On the Robustness of Safe Reinforcement Learning under Observational Perturbations

Zuxin Liu [1] **Zijian Guo** [1] **Zhepeng Cen** [1] **Huan Zhang** [2] **Jie Tan** [3] **Bo Li** [4] **Ding Zhao** [1]

Abstract

Safe reinforcement learning (RL) trains a policy to maximize the task reward while satisfying safety constraints. While prior works focus on the performance optimality, we find that the optimal solutions of many safe RL problems are not robust and safe against carefully designed observational perturbations. We formally analyze the unique properties of designing effective state adversarial attackers in the safe RL setting. We show that baseline adversarial attack techniques for standard RL tasks are not always effective for safe RL and proposed two new approaches - one maximizes the cost and the other maximizes the reward. One interesting and counter-intuitive finding is that the maximum reward attack is strong, as it can both induce unsafe behaviors and make the attack stealthy by maintaining the reward. We further propose a more effective adversarial training framework for safe RL and evaluate it via comprehensive experiments [1]. This work sheds light on the inherited connection between observational robustness and safety in RL and provides a pioneer work for future safe RL studies.

1. Introduction

Despite the great success of deep reinforcement learning in recent years (Mnih et al., 2013), it is still challenging to ensure safety when deploying them to safety-critical real-world applications. Safe reinforcement learning (RL) tackles the problem by solving a constrained optimization that can maximize the task reward while satisfying certain constraints (Garcıa & Fernández, 2015). This is usually done under the Constrained Markov Decision Processes (CMDPs) framework, and has shown to be effective in learning a constraint satisfaction policy in many tasks (Tessler et al., 2018). The success of recent safe RL approaches leverages the power of neural networks (Ray et al., 2019; Liu et al., 2022). However, it has been shown that neural networks are vulnerable to adversarial attacks – a small perturbation of the input data may lead to a large variance of the output (Machado et al., 2021; Pitropakis et al., 2019), which raises a concern when deploying a neural network RL policy to safety-critical applications (Akhtar & Mian, 2018). While many recent safe RL methods with deep policies can achieve outstanding constraint satisfaction performance in noise-free simulation environments, such a concern regarding their vulnerability under adversarial perturbations has not been studied in the safe RL setting. Particularly, we consider the observational perturbations that commonly exist in the physical world, such as unavoidable sensor errors and upstream perception inaccuracy (Zhang et al., 2020a).

Several recent works of observational robust RL have shown that deep RL agent could be attacked via sophistic observation perturbations, drastically decreasing their rewards (Huang et al., 2017; Zhang et al., 2020b). However, the robustness concept and adversarial training methods in standard RL setting may not be suitable for safe RL because of an additional metric that characterizes the cost of constraint violations (Brunke et al., 2021). The cost should be more important than the measure of reward, since any constrained violations could be fatal and unacceptable in the real world. For example, consider the autonomous vehicle navigation task where the reward is to reach the goal as fast as possible and the safety constraint is to not collide with obstacles, then sacrificing some reward is not comparable with violating the constraint because the latter one may cause catastrophic consequences. However, we find little research formally studying the robustness in the safe RL setting with adversarial observation perturbations, while we believe this should be an important aspect in the safe RL area, because **a vulnerable policy under adversarial attacks cannot be regarded as truly safe in the physical world.**

We aim to address the following questions in this work: 1) How vulnerable would a learned RL agent to adversarial attacks on its observations? 2) How to design effective at-

[1]Carnegie Mellon University [2]University of California, Los Angeles [3]Google Inc. [4]University of Illinois Urbana-Champaign. Correspondence to: Zuxin Liu <zuxinl@andrew.cmu.edu>, Ding Zhao <dingzhao@cmu.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]video demos are available at: https://sites.google.com/view/robustsaferl/home

tackers in the safe RL setting? 3) How to obtain a robust policy that can maintain safety even under worst-case perturbations? To answer the above questions, we formally define the observational robust safe RL problem and discuss the key metrics in evaluating the adversary and robustness of policy for safe RL. We also propose two strong adversarial attacks that can induce the agent performing unsafe behaviors, and show that adversarial training can help improve the robustness of constraint satisfaction. We summarize the contributions as follows.

1. As far as we are aware, we are the first to formally analyze the unique vulnerability of the optimal policy in safe RL under observational corruptions. We define the state-adversarial safe RL problem and investigate its fundamental properties. We show that the optimal solutions of safe RL problems are theoretically vulnerable under observational adversarial attacks.

2. We show that existing adversarial attack algorithms focusing on minimizing agent rewards do not always work, and propose two effective attack algorithms – one maximizes the constraint violation cost and one maximizes the reward. Surprisingly, the maximum reward attack is strong in inducing unsafe behaviors, both in theory and practically. We believe this property is overlooked as maximizing reward is the objective for standard RL, yet it leads to risky and stealthy attacks to safety constraints. Since this attack can maintain the nominal reward, it may not be detected in practice before catastrophic failures.

3. We propose an adversarial training algorithm with the proposed attackers and show contraction properties of their Bellman operators. Extensive experiments in continuous control tasks show that our method is more robust against adversarial perturbations in terms of constraint satisfaction.

2. Related Work

Safe RL. One type of approaches utilize domain knowledge of the target problem to improve the safety of an RL agent, such as designing a safety filter (Dalal et al., 2018), assuming a sophistic system dynamics model (Berkenkamp et al., 2017; Luo & Ma, 2021; Chen et al., 2021), or incorporating expert interventions (Saunders et al., 2017; Alshiekh et al., 2018). Constrained Markov Decision Process (CMDP) is another commonly used framework to model the safe RL problem, which can be solved via many constrained optimization techniques (Garcıa & Fernández, 2015). The Lagrangian-based method is a type of generic constrained optimization algorithm to solve CMDP, which introduces additional Lagrange multipliers to penalize constraints violations (Bhatnagar & Lakshmanan, 2012; Chow et al., 2017; Stooke et al., 2020). The multiplier can be optimized via gradient descend together with the policy parameters (Liang et al., 2018; Tessler et al., 2018), and can be easily incorporated in many existing unconstrained RL methods. Another line of work approximates the non-convex constrained optimization problem with low-order Taylor expansions, and then obtain the dual variable via convex optimization (Achiam et al., 2017; Yu et al., 2019; Zhang et al., 2020c; Yang et al., 2020). Since the constrained optimization-based methods are task-agnostic and more general, we will focus on the discussions of safe RL upon them.

Robust RL. The robustness definition in the RL context has many interpretations (Moos et al., 2022), including the robustness against action perturbations (Tessler et al., 2019), reward corruptions (Wang et al., 2020; Lin et al., 2020), domain shift (Tobin et al., 2017; Muratore et al., 2018), and dynamics uncertainty (Iyengar, 2005; Nilim & Ghaoui, 2003). The most related works are investigating the observational robustness of an RL agent under state adversarial attacks (Zhang et al., 2020a;b). It has been shown that the neural network policies can be easily attacked by adversarial observation noise and thus lead to much lower rewards than the optimal policy (Huang et al., 2017; Kos & Song, 2017; Lin et al., 2017; Pattanaik et al., 2017). However, most of the robust RL approaches model the attack and defense as a zero-sum game regarding the reward, while the robustness regarding safety, i.e., constraint satisfaction for safe RL, has not been formally investigated.

3. State Adversarial Attack for Safe RL

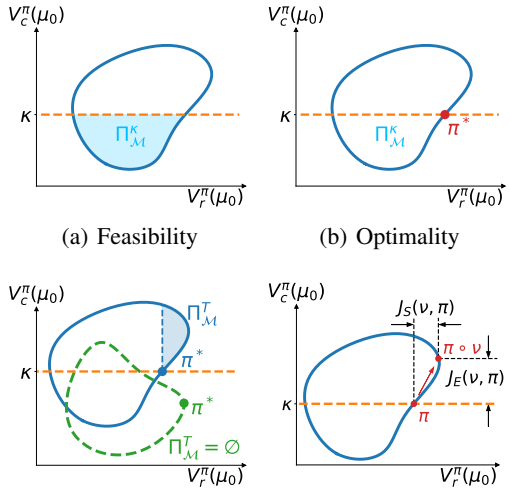
3.1. MDP, CMDP, and the safe RL problem

We consider an infinite horizon Markov Decision Process (MDP) that is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mu_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel that specifies the transition probability $p(s_{t+1}|s_t, a_t)$ from state s_t to s_{t+1} under the action a_t, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\gamma \to [0, 1)$ is the discount factor, and $\mu_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution. We consider the safe RL modeled under the Constrained Markov Decision Process (CMDP) framework (Altman, 1998), which augments the MDP tuple to $\mathcal{M} \coloneqq (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, c, \gamma, \mu_0)$ with an additional element $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, C_m]$ to characterize the cost for violating the constraint, where C_m is the maximum cost.

We denote a safe RL problem as $\mathcal{M}^{\kappa}_{\Pi} \coloneqq (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, c, \gamma, \mu_0, \Pi, \kappa)$, where Π is the policy class, and $\kappa \to [0, +\infty)$ is a threshold for constraint violation cost. Let $\pi(a|s) \in \Pi$ denote the policy and $\tau = \{s_0, a_0, ..., \}$ denote the trajectory. We use shorthand $f_t = f(s_t, a_t, s_{t+1}), f \in \{r, c\}$ for simplicity. The value function is $V^{\pi}_f(\mu_0) = \mathbb{E}_{\tau\sim\pi, s_0\sim\mu_0}[\sum_{t=0}^{\infty} \gamma^t f_t], f \in \{r, c\}$, which is the expectation of discounted return under the policy π and the initial state distribution μ_0. We overload the notation $V^{\pi}_f(s) = \mathbb{E}_{\tau\sim\pi, s_0=s}[\sum_{t=0}^{\infty} \gamma^t f_t], f \in \{r, c\}$ to denote the value function with the initial state $s_0 = s$, and denote $Q^{\pi}_f(s, a) = \mathbb{E}_{\tau\sim\pi, s_0=s, a_0=a}[\sum_{t=0}^{\infty} \gamma^t f_t], f \in \{r, c\}$

as the state-action value function under the policy π. The objective of $\mathcal{M}_\Pi^\kappa$ is to find the policy that maximizes the reward while limiting the cost incurred from constraint violations to a threshold κ:

$$\pi^* = \arg\max_\pi V_r^\pi(\mu_0), \quad s.t. \quad V_c^\pi(\mu_0) \leq \kappa. \tag{1}$$



(a) Feasibility (b) Optimality

(c) Temptation (d) Effectiveness and Stealthiness

Figure 1: Illustration of definitions via a mapping from the policy space to the metric plane $\Pi \to \mathbb{R}^2$, where the x-axis is the reward return and the y-axis is the cost return. A point on the metric plane denotes corresponding policies, i.e., the point (v_r, v_c) represents the policies $\{\pi \in \Pi | V_r^\pi(\mu_0) = v_r, V_c^\pi(\mu_0) = v_c\}$. The blue and green circles denote the policy space of two safe RL problems.

We then define feasibility, optimality and temptation to better describe the properties of a safe RL problem $\mathcal{M}_\Pi^\kappa$. The figure illustration of one example is shown in Fig. 1. Note that although the temptation concept naturally exists in many safe RL settings under the CMDP framework, we did not find formal descriptions or definitions of it in the literature.

Definition 3.1. Feasibility. The feasible policy class is the set of policies that satisfies the constraint with threshold κ: $\Pi_\mathcal{M}^\kappa := \{\pi(a|s) : V_c^\pi(\mu_0) \leq \kappa, \pi \in \Pi\}$. A feasible policy should satisfy $\pi \in \Pi_\mathcal{M}^\kappa$.

Definition 3.2. Optimality. A policy π^* is optimal in the safe RL context if 1) it is feasible: $\pi^* \in \Pi_\mathcal{M}^\kappa$; 2) no other feasible policy has higher reward return than it: $\forall \pi \in \Pi_\mathcal{M}^\kappa, V_r^{\pi^*}(\mu_0) \geq V_r^\pi(\mu_0)$.

We denote π^* as the optimal policy throughout the paper. Note that the optimality is defined w.r.t. the reward return within the feasible policy class $\Pi_\mathcal{M}^\kappa$ rather than the full policy class space Π, which means that policies that have higher reward return than π^* may exist in a safe RL problem due to the constraint, and we formally define them as tempting policies because they are rewarding but unsafe:

Definition 3.3. Temptation. We define the tempting policy class as the set of policies that have higher reward return than the optimal policy: $\Pi_\mathcal{M}^T := \{\pi(a|s) : V_r^\pi(\mu_0) > V_r^{\pi^*}(\mu_0), \pi \in \Pi\}$. **A tempting safe RL problem** has a non-empty tempting policy class: $\Pi_\mathcal{M}^T \neq \emptyset$.

We show that all the tempting polices are not feasible (proved by contradiction in Appendix A.1):

Lemma 3.4. *The tempting policy class and the feasible policy class are disjoint:* $\Pi_\mathcal{M}^T \cap \Pi_\mathcal{M}^\kappa = \emptyset$. *Namely, all the tempting policies violate the constraint:* $\forall \pi \in \Pi_\mathcal{M}^T, V_c^\pi(\mu_0) > \kappa$.

The existence of tempting policies is the unique feature and one of the major challenges of safe RL, since the agent need to update the policy carefully to prevent from being tempted when maximizing the reward. One can always tune the threshold κ to change the temptation status of a safe RL problem with the same CMDP. In this paper, we only consider the solvable **tempting** safe RL problems (i.e., the problems with a non-empty feasible class and a non-empty tempting class) because otherwise the non-tempting safe RL problem $\mathcal{M}_\Pi^\kappa$ can be reduced to a standard RL problem – an optimal policy could be obtained by maximizing the reward without considering the constraint, which is not the focus of this paper.

3.2. Safe RL under observational perturbations

We introduce a deterministic **observational** adversary $\nu(s) : \mathcal{S} \to \mathcal{S}$ which corrupts the state observation of the agent. We denote the corrupted state as $\tilde{s} := \nu(s)$ and the corrupted policy as $\pi \circ \nu := \pi(a|\tilde{s}) = \pi(a|\nu(s))$, as the state is first contaminated by ν and then used by the operator π. Note that the adversary does **not** modify the original CMDP and true states in the environment, but only the input of the agent. This setting mimics realistic scenarios, for instance, the adversary could be the noise from the sensing system or the errors from the upstream perception system.

Different from standard RL, safe RL has to ensure constraint satisfaction, since the cost of violating constraints in many safety-critical applications can be unaffordable. In addition, the reward metric is usually used to measure the agent's performance in finishing a task, so significantly reducing the task reward may warn the agent of the existence of attacks. As a result, a strong adversary in the safe RL setting aims to generate more constraint violations while maintaining high reward to make the attack stealthy (i.e., safety violation). In contrast, existing adversaries on standard RL aim to reduce the overall reward or lead to incorrect decision-making (i.e., observational robustness). This is one of our main contributions, which connects existing observational robustness with safety violation in RL. Concretely, we define two metrics to evaluate the adversary performance for safe RL:

Definition 3.5. (Attack) Effectiveness $J_E(\nu, \pi)$ is defined as the increased cost value under the adversary: $J_E(\nu, \pi) = V_c^{\pi \circ \nu}(\mu_0) - V_c^\pi(\mu_0)$. An adversary ν is effective if $J_E(\nu, \pi) > 0$.

The effectiveness metric measures an adversary's capability of attacking the safe RL agent to violate constraints. We also introduce another metric to characterize the adversary's stealthiness w.r.t. the task reward in the safe RL setting.

Definition 3.6. (Reward) Stealthiness $J_S(\nu, \pi)$ is defined as the increased reward value under the adversary: $J_S(\nu, \pi) = V_r^{\pi \circ \nu}(\mu_0) - V_r^{\pi}(\mu_0)$. An adversary ν is stealthy if $J_S(\nu, \pi) \geq 0$.

Note that the stealthiness concept is widely used in supervised learning (Sharif et al., 2016; Pitropakis et al., 2019). It usually refers to that the adversarial attack should be covert to human eyes regarding the input data so that it can hardly be identified (Machado et al., 2021). While the stealthiness regarding the state perturbation range is naturally satisfied based on the perturbation set definition, we introduce another level of stealthiness in terms of the task reward in the safe RL task. In some situations, a dramatic reward drop might be easily detected by the agent. A more stealthy attack is to maintain the task reward while increasing constraint violations, see Appendix B.1 for more discussions.

In practice, the power of the adversary is usually restricted (Madry et al., 2017; Zhang et al., 2020a), such that the perturbed observation will be limited within a pre-defined perturbation set $B(s)$: $\forall s \in \mathcal{S}, \nu(s) \in B(s)$. Following convention, we define the perturbation set $B_p^\epsilon(s)$ as the ℓ_p-ball around the original state: $\forall s' \in B_p^\epsilon(s), \|s' - s\|_p \leq \epsilon$, where ϵ is the ball size.

3.3. Adversarial attacks for safe RL

Given an optimal policy π^* of a tempting safe RL problem, we aim to design strong adversaries such that they are effective in making the agent unsafe and keep reward stealthiness in some applications. Motivated by Lemma 3.4, we propose the following **Maximum Reward (MR) attacker** that corrupts the observation of a policy π by maximizing the reward value:

$$\nu_{\text{MR}} = \arg\max_{\nu} V_r^{\pi \circ \nu}(\mu_0) \tag{2}$$

Proposition 3.7. *The MR attacker is guaranteed to be stealthy and effective for an optimal policy π^*, given enough large perturbation set $B_p^\epsilon(s)$ such that $V_r^{\pi^* \circ \nu_{MR}} > V_r^{\pi^*}$.*

The MR attacker is counter-intuitive because it is exactly the goal for standard RL.This is an interesting phenomenon worthy of highlighting, since we observe that the MR attacker is effective in making the optimal policy unsafe and retaining stealthy regarding the reward in the safe RL setting. The proof is given in Appendix A.1, which is based on the tempting policy property. We further observe the following important property for the optimal policy:

Lemma 3.8. *The optimal policy π^* of a tempting safe RL problem satisfies: $V_c^{\pi^*}(\mu_0) = \kappa$.*

Note that Lemma 3.8 holds in expectation rather than for a single trajectory. The proof is given in Appendix A.2. Lemma 3.8 suggest that the optimal policy in a tempting safe RL problem will be vulnerable as it is on the safety boundary, which motivates us to propose the **Maximum Cost (MC) attacker** that corrupts the observation of a policy π by maximizing the cost value:

$$\nu_{\text{MC}} = \arg\max_{\nu} V_c^{\pi \circ \nu}(\mu_0) \tag{3}$$

It is apparent to see that the MC attacker is effective w.r.t. the optimal policy given a large enough perturbation range, since we directly solve the adversarial state such that it can maximize the constraint violations. Therefore, as long as ν_{MC} can lead to a policy that has higher cost return than π^*, it is guaranteed to be effective in making the agent violate the constraint based on Lemma 3.8.

Practically, given a fixed policy π and its critics $Q_f^\pi(s, a), f \in \{r, c\}$, we obtain the corrupted state $\tilde{s}$ of s from the MR and MC attackers by solving:

$$\begin{aligned} \nu_{\text{MR}}(s) &= \arg\max_{\tilde{s} \in B_p^\epsilon(s)} \mathbb{E}_{\tilde{a} \sim \pi(a|\tilde{s})} \left[Q_r^\pi(s, \tilde{a}) \right] \\ \nu_{\text{MC}}(s) &= \arg\max_{\tilde{s} \in B_p^\epsilon(s)} \mathbb{E}_{\tilde{a} \sim \pi(a|\tilde{s})} \left[Q_c^\pi(s, \tilde{a}) \right] \end{aligned} \tag{4}$$

Suppose the policy π and the critics Q are all parametrized by differentiable models such as neural networks, then we can back-propagate the gradient through Q and π to solve the adversarial state $\tilde{s}$. This is similar to the policy optimization procedure in TD3 (Fujimoto et al., 2018) and DDPG (Lillicrap et al., 2015), whereas we replace the optimization domain from the policy parameter space to the observation space $B_p^\epsilon(s)$. The implementation details of the proposed attackers can be found in Appendix C.1.

3.4. Theoretical analysis of adversarial attacks

Theorem 3.9 (Existence of optimal and deterministic MC/MR attackers)**.** *A deterministic MC attacker ν_{MC} and a deterministic MR attacker ν_{MR} always exist, and there is no stochastic adversary ν' such that $V_c^{\pi \circ \nu'}(\mu_0) > V_c^{\pi \circ \nu_{MC}}(\mu_0)$ or $V_r^{\pi \circ \nu'}(\mu_0) > V_r^{\pi \circ \nu_{MR}}(\mu_0)$.*

Theorem 3.9 provides the theoretical foundation of Bellman operators that require optimal and deterministic adversaries in the next section. The proof is given in Appendix A.3. We can also obtain the upper-bound of constraint violations of the adversary attack at state s. Denote $\mathcal{S}_c$ as the set of unsafe states that have non-zero cost: $\mathcal{S}_c := \{s' \in \mathcal{S} : c(s, a, s') > 0\}$ and p_s as the maximum probability of entering unsafe states from state s: $p_s = \max_a \sum_{s' \in \mathcal{S}_c} p(s'|s, a)$.

Theorem 3.10 (One-step perturbation cost value bound)**.** *Suppose the optimal policy is locally L-Lipschitz continuous at state s: $D_{\text{TV}}[\pi(\cdot|s') \| \pi(\cdot|s)] \leq L \|s' - s\|_p$, and the perturbation set of the adversary $\nu(s)$ is an ℓ_p-ball $B_p^\epsilon(s)$. Let $\tilde{V}_c^{\pi,\nu}(s) = \mathbb{E}_{a \sim \pi(\cdot|\nu(s)), s' \sim p(\cdot|s,a)}[c(s, a, s') + \gamma V_c^\pi(s')]$ denote the cost value for only perturbing state s. The upper*

bound of $\tilde{V}_c^{\pi,\nu}(s)$ is given by:

$$\tilde{V}_c^{\pi,\nu}(s) - V_c^{\pi}(s) \leq 2L\epsilon \left(p_s C_m + \frac{\gamma C_m}{1-\gamma} \right). \quad (5)$$

Note that $\tilde{V}_c^{\pi,\nu}(s) \neq V_c^{\pi}(\nu(s))$ because the next state s' is still transited from the original state s, i.e., $s' \sim p(\cdot|s,a)$ instead of $s' \sim p(\cdot|\nu(s),a)$. Theorem 3.10 indicates that the power of an adversary is controlled by the policy smoothness L and perturbation range ϵ. In addition, the p_s term indicates that a safe policy should keep a safe distance to the unsafe state to prevent from being attacked. We further derive the upper bound of constraint violation for attacking the entire episodes.

Theorem 3.11 (Episodic perturbation cost value bound)**.** *Given a feasible policy $\pi \in \Pi_{\mathcal{M}}^{\kappa}$, suppose the L-Lipschitz continuity holds globally for π, and the perturbation set of ν is within an ℓ_p-ball, then the following bound holds:*

$$V_c^{\pi\circ\nu}(\mu_0) \leq \kappa + 2L\epsilon C_m \left(\frac{1}{1-\gamma} + \frac{4\gamma L\epsilon}{(1-\gamma)^2} \right) \left(\max_s p_s + \frac{\gamma}{1-\gamma} \right). \quad (6)$$

See Theorem 3.10, 3.11 proofs in Appendix A.4, A.5. We can still observe that the maximum cost value under perturbations is bounded by the Lipschitzness of the policy and the maximum perturbation range ϵ. The bound is tight since when $\epsilon \to 0$ (no attack) or $L \to 0$ (constant policy $\pi(\cdot|s)$ for all states), the RHS is 0 for Eq. (5) and κ for Eq. (6), which means that the attack is ineffective.

4. Observational Robust Safe RL

4.1. Adversarial training against observational perturbations

To defend against observational perturbations, we propose an adversarial safe RL training method. Similar to adversarial training in the supervised learning literature, we directly optimize the policy upon the attacked sampling trajectories $\tilde{\tau} = \{s_0, \tilde{a}_0, s_1, \tilde{a}_1, ...\}$, where $\tilde{a}_t \sim \pi(a|\nu(s_t))$. We can compactly represent the adversarial safe RL objective under observational perturbation as:

$$\pi^* = \arg\max_{\pi} V_r^{\pi\circ\nu}(\mu_0), \quad s.t. \quad V_c^{\pi\circ\nu}(\mu_0) \leq \kappa. \quad (7)$$

The key part is selecting proper adversaries during training to evaluate the value function under observational perturbations accurately, which can be done via the following Bellman operators, where $p_{sa}^{s'} = p(s'|s,a)$ and $f_{sa}^{s'} = f(s,a,s'), f \in \{r,c\}$:

Definition 4.1. Define the Bellman policy operator $\mathcal{T}_\pi$ as:

$$(\mathcal{T}_\pi V_f^{\pi\circ\nu})(s) = \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p_{sa}^{s'} \left[f_{sa}^{s'} + \gamma V_f^{\pi\circ\nu}(s') \right]. \quad (8)$$

Define the Bellman adversary effectiveness operator $\mathcal{T}_c^*$ as:

$$(\mathcal{T}_c^* V_c^{\pi\circ\nu})(s) = \max_{\tilde{s}\in B_p^\epsilon(s)} \sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p_{sa}^{s'} \left[c_{sa}^{s'} + \gamma V_c^{\pi\circ\nu}(s') \right]. \quad (9)$$

Define the Bellman adversary reward stealthiness operator $\mathcal{T}_r^*$ as :

$$(\mathcal{T}_r^* V_r^{\pi\circ\nu})(s) = \max_{\tilde{s}\in B_p^\epsilon(s)} \sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p_{sa}^{s'} \left[r_{sa}^{s'} + \gamma V_r^{\pi\circ\nu}(s') \right]. \quad (10)$$

The Bellman equation can be written as $V_f^{\pi\circ\nu}(s) = (\mathcal{T}_\pi V_f^{\pi\circ\nu})(s)$. We further prove the contraction properties:

Theorem 4.2 (Bellman contraction)**.** *The Bellman operators $\mathcal{T}_\pi, \mathcal{T}_c^*, \mathcal{T}_r^*$ are contractions under the sup-norm $\|\cdot\|_\infty$ and will converge to their fixed points, respectively. In addition, the fixed point for $\mathcal{T}_c^*$ is $V_c^{\pi\circ\nu_{MC}} = \mathcal{T}_c^* V_c^{\pi\circ\nu_{MC}}$, and the fixed point for $\mathcal{T}_r^*$ is $V_r^{\pi\circ\nu_{MR}} = \mathcal{T}_r^* V_r^{\pi\circ\nu_{MR}}$.*

Theorem 4.2 and 3.9 shows that we can evaluate the policy performance under a fixed deterministic adversary, which provides the theoretical justification of adversarial training, i.e., training a safe RL agent under observational perturbed sampling trajectories. In addition, the value functions under the MC and MR adversaries are the fixed points of the $\mathcal{T}_c^*$ and $\mathcal{T}_r^*$, which suggests that performing adversarial training for the RL agent with the MC and MR attackers will enable the agent to be robust against the most effective and the most reward stealthy perturbations, respectively. We have the following propositions:

Proposition 4.3. *Suppose an adversarial trained policy π' satisfies: $V_c^{\pi'\circ\nu_{MC}}(\mu_0) \leq \kappa$, then $\pi' \circ \nu$ is guaranteed to be feasible with any B_p^ϵ bounded adversarial perturbations.*

Proposition 4.3 indicates that by solving the adversarial constrained optimization problem under the MC attacker, all the feasible solutions will be safe under any bounded adversarial perturbations.

Proposition 4.4. *Suppose an adversarial trained policy π' satisfies: $V_c^{\pi'\circ\nu_{MR}}(\mu_0) \leq \kappa$, then $\pi' \circ \nu$ is guaranteed to be non-tempting with any B_p^ϵ bounded adversarial perturbations.*

Proposition 4.4 shows a nice property for training a robust policy, since the `max` operation over the reward in the safe RL objective may lead the policy to the tempting policy class, while the adversarial training with MR attacker can naturally keep the trained policy a safe distance from the tempting policy class, such that the adversarial trained policy will be robust against any bounded reward stealthy attackers. Practically, we observe that both MC and MR attackers can

increase the robustness and safety via adversarial training, and could be easily plugged in any safe RL algorithms, in principle.

4.2. Practical implementation

The meta adversarial training algorithm is shown in Algo. 1. We particularly adopt the primal-dual methods (Ray et al., 2019; Stooke et al., 2020; Tessler et al., 2018) that are widely used in the safe RL literature as the `learner`, then the adversarial training objective in Eq. (7) can be converted to a min-max form by using the Lagrange multiplier λ:

$$(\pi^*, \lambda^*) = \min_{\lambda \geq 0} \max_{\pi \in \Pi} V_r^{\pi \circ \nu}(\mu_0) - \lambda (V_c^{\pi \circ \nu}(\mu_0) - \kappa) \quad (11)$$

Solving the inner maximization (primal update) via any policy optimization methods and the outer minimization (dual update) via gradient descent iteratively yields the Lagrangian algorithm. Under proper learning rates and bounded noise assumptions, the iterates (π_n, λ_n) converge to a fixed point (a local minimum) almost surely (Tessler et al., 2018; Paternain et al., 2019). We will particularly use PPO (Schulman et al., 2017) in the primal update.

Algorithm 1 Adversarial safe RL training meta algorithm

Input: Policy class Π, Safe RL `learner`, Adversary `scheduler`
Output: Observational robust policy π

1: Initialize policy $\pi \in \Pi$ and adversary $\nu : \mathcal{S} \to \mathcal{S}$
2: **for** each training epoch $n = 1, ..., N$ **do**
3: Rollout trajectories: $\tilde{\tau} = \{s_0, \tilde{a}_0, ...\}_T$, $\tilde{a}_t \sim \pi(a|\nu(s_t))$
4: Run safe RL learner: $\pi \leftarrow$ `learner`$(\tilde{\tau}, \Pi)$
5: Update adversary: $\nu \leftarrow$ `scheduler`$(\tilde{\tau}, \pi, n)$
6: **end for**

Based on previous theoretical analysis, we adopt MC or MR as the adversary when sampling trajectories. The `scheduler` function aims to train the reward and cost Q-value functions for the MR and the MC attackers, because many on-policy algorithms such as PPO do not provide them. In addition, the scheduler can update the power of adversary based on the learning progress accordingly, since a strong adversary at the beginning may prohibit the `learner` exploring the environment and thus corrupts the training. We gradually increase the perturbation range ϵ along with the training epochs to adjust the adversary perturbation set B_p^ϵ, such that the agent will not be too conservative in the early stage of training. The similar idea is also used in adversarial training literature (Salimans et al., 2016; Arjovsky & Bottou, 2017; Gowal et al., 2018) and the curriculum learning literature (Dennis et al., 2020; Portelas et al., 2020). See more implementation details in Appendix C.3.

5. Experiment

In this section, we aim to answer the questions raised in Sec. 1. To this end, we adopt the robot locomotion continuous control tasks that are easy to interpret, motivated by safety, and used in many previous works (Achiam et al., 2017; Chow et al., 2019; Zhang et al., 2020c). The simulation environments are from a public available benchmark (Gronauer, 2022). We consider two tasks, and train multiple different robots (Car, Drone, Ant) for each task:

Run task. Agents are rewarded for running fast between two safety boundaries, and are given costs for violation constraints if they run across the boundaries or exceed an agent-specific velocity threshold. The tempting policies can violate the velocity constraint to obtain more rewards.

Circle task. The agents are rewarded for running in a circle in clock-wise direction, but are constrained to stay within a safe region that is smaller than the radius of the target circle. The tempting policies in this task will leave the safe region to gain more rewards.

We name each task via the `Robot-Task` format, for instance, `Car-Run`. In addition, we will use the PID PPO-Lagrangian (abbreviated as PPOL) method (Stooke et al., 2020) as the base safe RL algorithm to fairly compare different robust training approaches, while the proposed adversarial training can be used in other safe RL methods as well. The detailed hyperparameters of the adversaries and safe RL algorithms can be found in Appendix C.

5.1. Adversarial attacker comparison

We first demonstrate the vulnerability of the optimal safe RL policies without adversarial training and compare the performance of different adversaries. All the adversaries have the same ℓ_∞ norm perturbation set B_∞^ϵ restriction. We adopt three adversary baselines, including one improved version:

Random attacker baseline. This is a simple baseline by sampling the corrupted observations randomly within the perturbation set via a uniform distribution.

Maximum Action Difference (MAD) attacker baseline. The MAD attacker (Zhang et al., 2020a) is designed for standard RL tasks, which is shown to be effective in decreasing a trained RL agent's reward return. The optimal adversarial observation is obtained by maximizing the KL-divergence between the corrupted policy:

$$\nu_{\text{MAD}}(s) = \arg\max_{\tilde{s} \in B_p^\epsilon(s)} D_{\text{KL}} \left[\pi(a|\tilde{s}) \| \pi(a|s)\right]$$

Adaptive MAD (AMAD) attacker baseline. Since the vanilla MAD attacker is not designed for safe RL, we further improve it to an adaptive version as a stronger baseline. The motivation comes from Lemma 3.8 – the optimal policy will be close to the constraint boundary that with high risks (see Appendix C.6 for more details): Therefore, AMAD only perturbs the observation when the agent is within high-risk regions that is determined by the cost value function and a threshold ξ to achieve more effective attack:

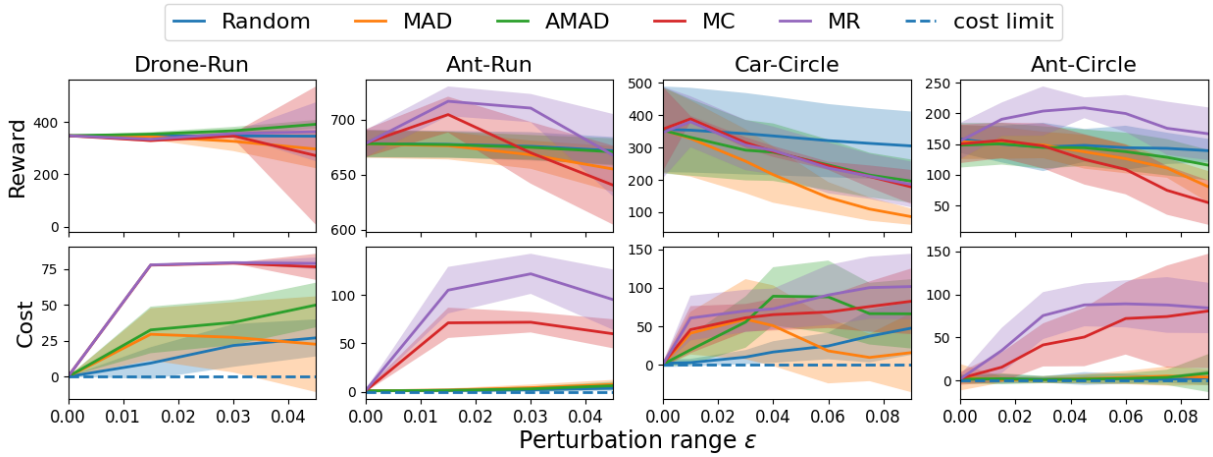


Figure 2: Reward and cost curves of all 5 attackers evaluated on well-trained vanilla PPO-Lagrangian models w.r.t. the perturbation range ϵ. The curves are averaged over 50 episodes and 5 seeds, where the solid lines are the mean and the shadowed areas are the standard deviation. The dashed line is the cost without perturbations.

$$\nu_{\text{AMAD}}(s) := \begin{cases} \nu_{\text{MAD}}(s), & \text{if } V_c^{\pi}(s) \geq \xi, \\ s, & \text{otherwise .} \end{cases}$$

Experiment setting. We evaluate the performance of all three baselines above and our MC, MR adversaries by attacking well-trained PPO-Lagrangian policies in different tasks. The trained policies can achieve nearly zero constraint violation costs without observational perturbations. We keep the trained model weights and environment seeds fixed for all the attackers to ensure fair comparison.

Experiment result. Fig. 2 shows the attack results of the 5 adversaries on PPOL-vanilla. Each column corresponds to an environment. The first row is the episode reward and the second row is the episode cost of constraint violations. We can see that the vanilla safe RL policies are vulnerable, since the safety performance deteriorates (cost increases) significantly even with a small adversarial perturbation range ϵ. Generally, we can see an increasing cost trend as the ϵ increases, except the MAD attacker. Although MAD can reduce the agent's reward quite well, it fails to perform an effective attack in increasing the cost because the reward decrease may keep the agent away from high-risk regions. It is even worse than the random attacker in the `Car-Circle` task. The improved AMAD attacker is a stronger baseline than MAD, as it only attacks in high-risk regions and thus has a higher chance of entering unsafe regions to induce more constraint violations. More comparisons between MAD and AMAD can be found in Appendix C.9. Our proposed MC and MR attackers outperform all baselines attackers (Random, MAD and AMAD) in terms of effectiveness by increasing the cost with a large margin in most tasks. Surprisingly, the MR attacker can achieve even higher costs than MC and is more stealthy as it can maintain or increase the reward well, which validates our theoretically analysis and the existence of tempting policies.

5.2. Performance of safe RL with Adversarial Training

We adopt 5 baselines, including the **PPOL-vanilla** method without robust training, the naive adversarial training under random noise **PPOL-random**, the state-adversarial algorithm **SA-PPOL** that is proposed in (Zhang et al., 2020a), but we extend their PPO in standard RL setting to PPOL in the safe RL setting. The original SA-PPOL algorithm utilizes the MAD attacker to compute the adversarial states, and then adds a KL regularizer to penalize the divergence between them and the original states. We add two additional baselines **SA-PPOL(MC)** and **SA-PPOL(MR)** for ablation study, where we change the MAD attacker to our proposed MC and MR adversaries. Our adversarial training methods are named as **ADV-PPOL(MC)** and **ADV-PPOL(MR)**, which are trained under the MC and MR attackers respectively. We use the same PPOL implementation and hyperparameters for all methods for fair comparison. More details of the baselines and hyperparameters can be found in Appendix C.5-C.8.

Results. The evaluation results of different trained policies under adversarial attacks are shown in Table 1, where **Natural** represents the performance without noise. We train each algorithm with 5 random seeds and evaluate each trained policy with 50 episodes under each attacker to obtain the values. The training and testing perturbation ranges ϵ are the same. We use gray shadows to highlight the top two safest agents that with the smallest cost values, but we ignore the failure agents whose rewards are less than 50% of the PPOL-vanilla method. We mark the failure agents with $\star$. Due to the page limit, we leave the evaluation results under random and MAD attackers to Appendix C.9.

Analysis. We can observe that although most baselines can

Table 1: Evaluation results of natural performance (no attack) and under 3 attackers. Our methods are ADV-PPOL(MC/MR). Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds. We shadow two lowest-costs agents under each attacker column and break ties based on rewards, excluding the failing agents (whose natural rewards are less than 50% of PPOL-vanilla's). We mark the failing agents with ⋆.

Env	Method	Natural		AMAD		MC		MR	
		Reward	Cost	Reward	Cost	Reward	Cost	Reward	Cost
Car-Run $\epsilon = 0.05$	PPOL-vanilla	561.33±1.97	0.15±0.36	548.04±17.72	13.49±30.93	624.49±8.87	184.09±0.52	624.46±8.86	184.06±0.47
	PPOL-random	556.84±1.87	0.01±0.08	550.66±1.71	1.73±2.05	584.18±2.69	183.79±0.73	585.79±2.01	183.91±0.59
	SA-PPOL	545.87±2.13	0.0±0.0	546.95±2.11	0.0±0.0	571.14±1.5	6.44±7.28	571.07±1.33	1.77±3.31
	SA-PPOL(MC)	552.58±3.84	0.0±0.0	541.58±3.6	0.0±0.0	568.96±1.92	1.17±2.07	569.6±1.51	0.67±1.24
	SA-PPOL(MR)	543.0±1.14	0.0±0.0	537.19±1.47	0.0±0.0	568.32±1.95	16.27±23.2	568.29±2.12	12.23±17.26
	ADV-PPOL(MC)	525.76±2.99	0.0±0.0	516.22±3.52	0.0±0.0	555.64±3.44	0.05±0.21	554.48±2.78	0.01±0.08
	ADV-PPOL(MR)	525.93±2.28	0.0±0.0	514.97±2.68	0.0±0.0	557.38±2.83	0.06±0.24	556.87±3.03	0.05±0.22
Drone-Run $\epsilon = 0.025$	PPOL-vanilla	347.17±1.53	0.0±0.0	362.18±11.24	35.69±17.74	336.05±9.43	79.0±0.0	345.64±5.22	79.0±0.0
	PPOL-random	343.71±1.55	0.0±0.0	361.85±18.03	65.58±22.58	268.28±4.26	0.9±2.28	317.25±40.34	33.66±29.86
	SA-PPOL	284.47±32.13	0.0±0.0	306.55±26.44	11.85±18.04	156.97±384.11	61.88±18.49	403.11±163.75	75.73±7.71
	⋆ SA-PPOL(MC)	174.61±34.37	0.06±0.73	86.35±56.22	0.0±0.0	205.34±29.57	10.13±10.67	217.51±24.77	8.31±6.05
	⋆ SA-PPOL(MR)	0.13±0.22	0.0±0.0	0.11±0.21	0.0±0.0	0.25±0.37	0.0±0.0	0.28±0.43	0.0±0.0
	ADV-PPOL(MC)	273.4±16.98	0.0±0.0	268.0±12.0	0.05±0.57	275.0±28.26	1.1±3.17	294.8±23.67	18.11±25.87
	ADV-PPOL(MR)	233.31±20.68	0.0±0.0	238.0±22.15	0.0±0.0	229.8±68.0	6.81±8.38	238.11±46.17	0.95±1.92
Ant-Run $\epsilon = 0.025$	PPOL-vanilla	678.4±12.64	1.23±1.4	676.23±12.27	2.68±2.16	661.3±58.17	66.41±14.07	706.32±18.83	112.33±25.57
	PPOL-random	673.42±14.47	1.01±1.06	670.6±13.59	1.9±1.47	661.47±10.02	45.94±10.2	670.85±18.73	46.97±11.63
	SA-PPOL	658.83±14.14	0.46±0.82	658.42±13.96	0.66±0.87	668.14±25.7	67.68±20.17	694.86±11.05	87.1±20.78
	SA-PPOL(MC)	574.36±25.69	3.03±3.15	574.85±26.37	3.16±3.48	604.77±30.51	21.39±10.83	619.4±31.35	32.87±12.69
	⋆ SA-PPOL(MR)	90.49±60.14	5.33±4.27	77.64±84.93	5.17±4.24	77.99±72.79	6.33±4.87	69.93±96.51	6.17±4.87
	ADV-PPOL(MC)	601.25±18.6	0.0±0.0	599.31±18.34	0.0±0.0	666.73±15.21	1.1±1.02	665.47±18.29	1.75±1.54
	ADV-PPOL(MR)	620.17±27.28	0.17±0.41	618.04±24.66	0.31±0.55	634.96±14.94	4.07±2.35	648.95±17.67	4.69±2.81
Car Circle $\epsilon = 0.05$	PPOL-vanilla	337.69±152.34	1.8±3.91	274.61±78.92	92.53±39.32	265.61±12.43	69.33±18.91	238.06±101.01	74.47±38.44
	PPOL-random	398.71±48.96	0.17±0.9	293.77±105.83	69.97±46.75	307.77±32.95	59.3±26.29	295.2±45.56	49.73±22.98
	SA-PPOL	403.92±22.63	0.4±2.15	382.8±20.57	0.37±1.97	361.0±12.96	109.1±6.0	452.98±26.72	89.03±9.13
	SA-PPOL(MC)	417.78±17.79	0.33±1.8	314.13±27.73	0.0±0.0	355.12±13.42	98.43±14.52	468.1±14.41	87.5±9.17
	SA-PPOL(MR)	389.03±47.53	0.2±1.0	351.49±34.16	0.14±0.69	342.67±39.23	77.9±21.57	414.87±66.09	75.68±20.73
	ADV-PPOL(MC)	302.3±12.24	0.1±0.7	296.23±19.02	1.86±5.49	310.37±25.68	1.12±3.98	261.52±24.51	0.28±1.59
	ADV-PPOL(MR)	309.42±35.45	0.0±0.0	321.44±20.52	6.66±10.94	258.52±31.53	0.08±0.56	308.6±54.7	0.16±1.12
Drone Circle $\epsilon = 0.025$	PPOL-vanilla	627.49±55.24	0.27±1.12	527.6±171.54	34.5±36.73	228.79±181.92	95.17±59.23	85.79±159.67	174.4±81.58
	PPOL-random	604.31±46.83	0.37±1.97	559.2±173.25	27.67±32.66	159.16±184.15	91.5±98.26	130.08±146.67	103.1±92.03
	SA-PPOL	503.13±19.89	0.0±0.0	496.34±20.54	0.0±0.0	430.64±89.64	97.57±27.47	346.99±320.08	109.5±78.1
	SA-PPOL(MC)	347.43±97.49	8.5±35.32	346.25±41.68	0.0±0.0	329.05±143.47	58.77±34.94	380.53±176.19	78.07±60.05
	⋆ SA-PPOL(MR)	184.7±128.7	11.94±43.67	189.76±118.14	15.38±47.38	189.18±142.46	44.62±35.83	219.87±138.35	49.14±52.87
	ADV-PPOL(MC)	359.02±33.01	0.0±0.0	351.57±52.5	1.48±6.44	399.78±69.47	4.16±12.57	356.09±90.42	9.66±28.48
	ADV-PPOL(MR)	356.6±46.91	0.0±0.0	339.04±72.43	5.36±23.08	275.43±95.08	5.66±22.41	379.52±87.22	1.2±6.47
Ant Circle $\epsilon = 0.025$	PPOL-vanilla	157.44±26.21	2.7±6.02	143.37±36.86	3.23±9.87	153.98±34.52	38.93±29.78	208.81±20.1	70.53±22.6
	PPOL-random	155.81±16.84	2.67±6.6	150.65±17.63	2.17±5.02	114.24±35.22	1.83±6.08	183.07±24.63	58.53±22.3
	SA-PPOL	143.34±32.08	0.13±0.56	142.66±35.01	4.53±10.67	159.02±43.95	37.47±26.5	203.85±27.56	51.47±27.79
	⋆ SA-PPOL(MC)	-0.62±1.72	0.0±0.0	0.09±1.27	0.0±0.0	-0.17±1.46	0.0±0.0	-0.34±1.61	0.0±0.0
	⋆ SA-PPOL(MR)	-0.8±2.28	0.0±0.0	-0.57±2.2	0.0±0.0	-0.89±2.12	0.0±0.0	-0.86±2.09	0.0±0.0
	ADV-PPOL(MC)	135.98±15.99	0.3±1.62	130.76±18.87	0.77±4.13	137.13±29.4	6.33±13.96	134.68±22.01	5.3±9.39
	ADV-PPOL(MR)	133.27±19.53	0.87±3.25	127.19±32.64	1.2±4.49	118.57±26.37	0.83±2.02	141.74±23.63	1.07±3.08

achieve near zero natural cost, their safety performances are vulnerable under the strong MC and MR attackers, which are more effective than AMAD in inducing unsafe behaviors. The proposed adversarial training methods (ADV-PPOL) consistently outperform baselines in safety with the lowest costs while maintaining high rewards in most tasks. The comparison with PPOL-random indicates that the MC and MR attackers are essential ingredients of adversarial training. Although SA-PPOL agents can maintain reward very well, they are not safe as to constraint satisfaction under adversarial perturbations in most environments. The ablation studies with SA-PPOL(MC) and SA-PPOL(MC) suggest that the KL-regularized robust training technique, which is successful in standard robust RL setting, does not work well for safe RL even with the same adversarial attacks during training, and they may also fail to obtain a high-rewarding policy in some tasks (see discussions of the training failure in Appendix B.2). As a result, we can conclude that the proposed adversarial training methods with the MC and MR attackers are better than baselines regarding both training stability and testing robustness and safety.

6. Conclusion

We study the observational robustness regarding constraint satisfaction for safe RL and show that the optimal policy of tempting safe RL problems could be vulnerable. We propose two effective attackers to induce unsafe behaviors. An interesting and surprising finding is that maximizing the reward attack is as effective as directly maximizing the cost while keeping stealthiness. We further propose an adversarial training method to increase the robustness and safety performance for safe RL, and a wide range of experiments show that the proposed method outperforms the robust training techniques for standard RL settings. We hope this work can attract more attention in the safe RL community to studying safety from the robustness perspective, as both safety and robustness are important ingredients before deploying RL to the real world.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.

Akhtar, N. and Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Altman, E. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical methods of operations research*, 48(3):387–417, 1998.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. Safe model-based reinforcement learning with stability guarantees. *arXiv preprint arXiv:1705.08551*, 2017.

Bhatnagar, S. and Lakshmanan, K. An online actor–critic algorithm with function approximation for constrained markov decision processes. *Journal of Optimization Theory and Applications*, 153(3):688–708, 2012.

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A. P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5, 2021.

Chen, B., Liu, Z., Zhu, J., Xu, M., Ding, W., and Zhao, D. Context-aware safe reinforcement learning for non-stationary environments. *arXiv preprint arXiv:2101.00531*, 2021.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

Chow, Y., Nachum, O., Faust, A., Duenez-Guzman, E., and Ghavamzadeh, M. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., and Levine, S. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in Neural Information Processing Systems*, 33: 13049–13061, 2020.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Gronauer, S. Bullet-safety-gym: Aframework for constrained reinforcement learning. 2022.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Kakade, S. M. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Kos, J. and Song, D. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Liang, Q., Que, F., and Modiano, E. Accelerated primal-dual policy optimization for safe reinforcement learning. *arXiv preprint arXiv:1802.06480*, 2018.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

Lin, Z., Thomas, G., Yang, G., and Ma, T. Model-based adversarial meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 33:10161–10173, 2020.

Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, Z. S., Li, B., and Zhao, D. Constrained variational policy optimization for safe reinforcement learning. *arXiv preprint arXiv:2201.11927*, 2022.

Luo, Y. and Ma, T. Learning barrier certificates: Towards safe reinforcement learning with zero training-time violations. *Advances in Neural Information Processing Systems*, 34, 2021.

Machado, G. R., Silva, E., and Goldschmidt, R. R. Adversarial machine learning in image classification: A survey toward the defender's perspective. *ACM Computing Surveys (CSUR)*, 55(1):1–38, 2021.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Meir, A. and Keeler, E. A theorem on contraction mappings. *Journal of Mathematical Analysis and Applications*, 28 (2):326–329, 1969.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., and Peters, J. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. G. Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*, 2016.

Muratore, F., Treede, F., Gienger, M., and Peters, J. Domain randomization for simulation-based policy optimization with transferability assessment. In *Conference on Robot Learning*, pp. 700–713. PMLR, 2018.

Nilim, A. and Ghaoui, L. Robustness in markov decision problems with uncertain transition matrices. *Advances in neural information processing systems*, 16, 2003.

Paternain, S., Chamon, L. F., Calvo-Fullana, M., and Ribeiro, A. Constrained reinforcement learning has zero duality gap. *arXiv preprint arXiv:1910.13393*, 2019.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

Pitropakis, N., Panaousis, E., Giannetsos, T., Anastasiadis, E., and Loukas, G. A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34: 100199, 2019.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

Ray, A., Achiam, J., and Amodei, D. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7, 2019.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Saunders, W., Sastry, G., Stuhlmueller, A., and Evans, O. Trial without error: Towards safe reinforcement learning via human intervention. *arXiv preprint arXiv:1707.05173*, 2017.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pp. 1528–1540, 2016.

Stooke, A., Achiam, J., and Abbeel, P. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Sutton, R. S., Barto, A. G., et al. Introduction to reinforcement learning. 1998.

Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Tessler, C., Efroni, Y., and Mannor, S. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pp. 6215–6224. PMLR, 2019.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Wang, J., Liu, Y., and Li, B. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6202–6209, 2020.

Yang, T.-Y., Rosca, J., Narasimhan, K., and Ramadge, P. J. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

Yu, M., Yang, Z., Kolar, M., and Wang, Z. Convergent policy optimization for safe reinforcement learning. *arXiv preprint arXiv:1910.12156*, 2019.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., and Hsieh, C.-J. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33: 21024–21037, 2020a.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., and Hsieh, C.-J. Robust deep reinforcement learning against adversarial perturbations on state observations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21024–21037. Curran Associates, Inc., 2020b. URL https://proceedings.neurips.cc/paper/2020/file/f0eb6568ea114ba6e293f903c34d7488-Paper.pdf.

Zhang, Y., Vuong, Q., and Ross, K. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 2020c.

A. Proofs and Discussions

A.1. Proof of Lemma 3.4 and Proposition 3.7 – infeasible tempting policies

Lemma 3.4 indicates that all the tempting policies are infeasible: $\forall \pi \in \Pi_{\mathcal{M}}^{T}, V_c^{\pi}(\mu_0) > \kappa$. We will prove it by contradiction.

Proof. For a tempting safe RL problem $\mathcal{M}_{\Pi}^{\kappa}$, there exists a tempting policy that satisfies the constraint: $\pi' \in \Pi_{\mathcal{M}}^{T}, V_c^{\pi'}(\mu_0) \leq \kappa, \pi' \in \Pi_{\mathcal{M}}^{\kappa}$. Denote the optimal policy as π^*, then based on the definition of the tempting policy, we have $V_r^{\pi'}(\mu_0) > V_r^{\pi^*}(\mu_0)$. Based on the definition of optimality, we know that for any other feasible policy $\pi \in \Pi_{\mathcal{M}}^{\kappa}$, we have:

$$V_r^{\pi'}(\mu_0) > V_r^{\pi^*}(\mu_0) \geq V_r^{\pi}(\mu_0),$$

which indicates that π' is the optimal policy for $\mathcal{M}_{\Pi}^{\kappa}$. Then again, based on the definition of tempting policy, we will obtain:

$$V_r^{\pi'}(\mu_0) > V_r^{\pi'}(\mu_0),$$

which contradicts to the fact that $V_r^{\pi'}(\mu_0) = V_r^{\pi'}(\mu_0)$. Therefore, there is no tempting policy that satisfies the constraint. □

Proposition 3.7 suggest that as long as the MR attacker can successfully obtain a policy that has higher reward return than the optimal policy π^* given enough large perturbation set $B_p^{\epsilon}(s)$, it is guaranteed to be reward stealthy and effective.

Proof. The stealthiness is naturally satisfied based on the definition. The effectiveness is guaranteed by Lemma 3.4. Since the corrupted policy $\pi^* \circ \nu_{\text{MR}}$ can achieve $V_r^{\pi^* \circ \nu_{\text{MR}}} > V_r^{\pi^*}$, we can conclude that $\pi^* \circ \nu_{\text{MR}}$ is within the tempting policy class, since it has higher reward than the optimal policy. Then we know that it will violate the constraint based on Lemma 3.4, and thus the MR attacker is effective. □

A.2. Proof of Lemma 3.8 – optimal policy's cost value

Lemma 3.8 says that the optimal policy π^* of a tempting safe RL problem satisfies: $V_c^{\pi^*}(\mu_0) = \kappa$. We will also prove it by contradiction.

Proof. Suppose the optimal policy π^* for a tempting safe RL problem $\mathcal{M}_{\Pi}^{\kappa}$ has: $V_c^{\pi^*}(\mu_0) < \kappa$. Denote $\pi' \in \Pi_{\mathcal{M}}^{T}$ as a tempting policy. Based on Lemma 3.4, we know that $V_c^{\pi'}(\mu_0) > \kappa$ and $V_r^{\pi'}(\mu_0) > V_r^{\pi^*}(\mu_0)$. Then we can compute a weight α:

$$\alpha = \frac{\kappa - V_c^{\pi^*}(\mu_0)}{V_c^{\pi'}(\mu_0) - V_c^{\pi^*}(\mu_0)}. \tag{12}$$

We can see that:

$$\alpha V_c^{\pi'}(\mu_0) + (1-\alpha) V_c^{\pi^*}(\mu_0) = \kappa. \tag{13}$$

We further define another policy $\bar{\pi}$ based on the mixture of π^* and π', such that a trajectory of a whole episode has α probability to be sampled from π' and $1-\alpha$ probability to be sampled from π^*:

$$\tau \sim \bar{\pi} := \begin{cases} \tau \sim \pi', & \text{with probability } \alpha, \\ \tau \sim \pi^*, & \text{with probability } 1-\alpha. \end{cases} \tag{14}$$

Then we can conclude that $\bar{\pi}$ is also feasible:

$$V_c^{\bar{\pi}}(\mu_0) = \mathbb{E}_{\tau\sim\bar{\pi}}[\sum_{t=0}^{\infty} \gamma^t c_t] = \alpha \mathbb{E}_{\tau\sim\pi'}[\sum_{t=0}^{\infty} \gamma^t c_t] + (1-\alpha)\mathbb{E}_{\tau\sim\pi^*}[\sum_{t=0}^{\infty} \gamma^t c_t] \tag{15}$$

$$= \alpha V_c^{\pi'}(\mu_0) + (1-\alpha) V_c^{\pi^*}(\mu_0) = \kappa. \tag{16}$$

In addition, $\bar{\pi}$ has higher reward return than the optimal policy π^*:

$$V_r^{\bar{\pi}}(\mu_0) = \mathbb{E}_{\tau\sim\bar{\pi}}[\sum_{t=0}^{\infty}\gamma^t r_t] = \alpha\mathbb{E}_{\tau\sim\pi'}[\sum_{t=0}^{\infty}\gamma^t r_t] + (1-\alpha)\mathbb{E}_{\tau\sim\pi^*}[\sum_{t=0}^{\infty}\gamma^t r_t] \tag{17}$$

$$= \alpha V_r^{\pi'}(\mu_0) + (1-\alpha)V_r^{\pi^*}(\mu_0) \tag{18}$$

$$> \alpha V_r^{\pi^*}(\mu_0) + (1-\alpha)V_r^{\pi^*}(\mu_0) = V_r^{\pi^*}(\mu_0), \tag{19}$$

where the inequality comes from the definition of the tempting policy. Since $\bar{\pi}$ is both feasible, and has strictly higher reward return than the policy π^*, we know that π^* is not optimal, which contradicts to our assumption. Therefore, the optimal policy π^* should always satisfy $V_c^{\pi^*}(\mu_0) = \kappa$.

□

Remark A.1. The cost value function $V_c^{\pi^*}(\mu_0) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t c_t]$ is based on the expectation of the sampled trajectories (expectation over episodes) rather than a single trajectory (expectation within one episode), because for a single sampled trajectory $\tau\sim\pi$, $V_c^{\pi^*}(\tau) = \sum_{t=0}^{\infty}\gamma^t c_t$ may even not necessarily satisfy the constraint.

Remark A.2. The proof also indicates that the range of metric function $\mathcal{V} := \{(V_r^{\pi}(\mu_0), V_c^{\pi}(\mu_0))\}$ (as shown as the blue circle in Fig.1) is convex when enlarging the domain from Π to a linear policy mixture space $\bar{\Pi}$. For simplicity, we define $\langle\boldsymbol{\alpha}, \boldsymbol{\pi}\rangle$ as a policy mixture $\bar{\pi}\in\bar{\Pi}$ which samples mixed trajectories episodically,

$$\tau\sim\langle\boldsymbol{\alpha}, \boldsymbol{\pi}\rangle := \tau\sim\pi_i, \text{ with probability } \alpha_i, i = 1, 2, \dots, \tag{20}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots], \alpha_i \geq 0, \sum_{i=1}\alpha_i = 1, \boldsymbol{\pi} = [\pi_1, \pi_2, \dots]$. Similar to the above proof, we have

$$V_f^{\langle\boldsymbol{\alpha},\boldsymbol{\pi}\rangle}(\mu_0) = \langle\boldsymbol{\alpha}, V_f^{\boldsymbol{\pi}}(\mu_0)\rangle, f\in\{r, c\}, \tag{21}$$

where $V_f^{\boldsymbol{\pi}}(\mu_0) = [V_f^{\pi_1}(\mu_0), V_f^{\pi_2}(\mu_0), \dots]$. Consider $\forall(v_{r1}, v_{c1}), (v_{r2}, v_{c2})\in\mathcal{V}$, suppose they correspond to policy mixture $\langle\boldsymbol{\alpha}, \boldsymbol{\pi}\rangle$ and $\langle\boldsymbol{\beta}, \boldsymbol{\pi}\rangle$ respectively, then $\forall t\in[0, 1]$, the new mixture $\langle t\boldsymbol{\alpha} + (1-t)\boldsymbol{\beta}, \boldsymbol{\pi}\rangle \in \bar{\Pi}$ and $V_f^{\langle t\boldsymbol{\alpha}+(1-t)\boldsymbol{\beta},\boldsymbol{\pi}\rangle}(\mu_0) = t\cdot v_{f1} + (1-t)\cdot v_{f2}\in\mathcal{V}$. Therefore, $\mathcal{V}$ is a convex set.

A.3. Proof of Theorem 3.9 – existence of optimal deterministic MC/MR adversary

Existence. Given a fixed policy π, We first introduce two adversary MDPs $\hat{\mathcal{M}}_r = (\mathcal{S}, \hat{\mathcal{A}}, \hat{\mathcal{P}}, \hat{R}_r, \gamma)$ for reward maximization adversary and $\hat{\mathcal{M}}_c = (\mathcal{S}, \hat{\mathcal{A}}, \hat{\mathcal{P}}, \hat{R}_c, \gamma)$ for cost maximization adversary to prove the existence of optimal adversary. In adversary MDPs, the adversary acts as the agent to choose a perturbed state as the action (i.e., $\hat{a} = \tilde{s}$) to maximize the cumulative reward $\sum\hat{R}$. Therefore, in adversary MDPs, the action space $\hat{\mathcal{A}} = \mathcal{S}$ and $\nu(\cdot|s)$ denotes a policy distribution.

Based on the above definitions, we can also derive transition function and reward function for new MDPs (Zhang et al., 2020a)

$$\hat{p}(s'|s, a) = \sum_a \pi(a|\hat{a})p(s'|s, a), \tag{22}$$

$$\hat{R}_f(s, \hat{a}, s') = \begin{cases} \frac{\sum_a \pi(a|\hat{a})p(s'|s,a)f(s,a,s')}{\sum_a \pi(a|\hat{a})p(s'|s,a)}, & \hat{a}\in B_p^{\epsilon}(s) \\ -C, & \hat{a}\notin B_p^{\epsilon}(s) \end{cases}, f\in\{r, c\}, \tag{23}$$

where $\hat{a} = \tilde{s}\sim\nu(\cdot|s)$ and C is a constant. Therefore, with sufficiently large C, we can guarantee that the optimal adversary ν^* will not choose a perturbed state $\hat{a}$ out of the l_p-ball of the given state s, i.e., $\nu^*(\hat{a}|s) = 0, \forall\hat{a}\notin B_p^{\epsilon}(s)$.

According to the properties of MDP (Sutton et al., 1998), $\hat{\mathcal{M}}_r, \hat{\mathcal{M}}_c$ have corresponding optimal policy ν_r^*, ν_c^*, which are deterministic by assigning unit mass probability to the optimal action $\hat{a}$ for each state.

Next, we will prove that $\nu_r^* = \nu_{\text{MR}}, \nu_c^* = \nu_{\text{MC}}$. Consider value function in $\hat{\mathcal{M}}_f, f\in\{r, c\}$, for an adversary $\nu\in\mathcal{N} :=$

$\{\nu|\nu^*(\hat{a}|s) = 0, \forall \hat{a} \notin B_p^\epsilon(s)\}$, we have

$$\hat{V}_f^\nu(s) = \mathbb{E}_{\hat{a}\sim\nu(\cdot|s),s'\sim\hat{p}(\cdot|s,\hat{a})}[\hat{R}_f(s,\hat{a},s') + \gamma\hat{V}_f^\nu(s')] \tag{24}$$

$$= \sum_{\hat{a}} \nu(\hat{a}|s) \sum_{s'} \hat{p}(s'|s,\hat{a})[\hat{R}_f(s,\hat{a},s') + \gamma\hat{V}_f^\nu(s')] \tag{25}$$

$$= \sum_{\hat{a}} \nu(\hat{a}|s) \sum_{s'} \sum_{a} \pi(a|\hat{a})p(s'|s,a) \left[\frac{\sum_a \pi(a|\hat{a})p(s'|s,a)f(s,a,s')}{\sum_a \pi(a|\hat{a})p(s'|s,a)} + \gamma\hat{V}_f^\nu(s')\right] \tag{26}$$

$$= \sum_{s'} p(s'|s,a) \sum_{a} \pi(a|\hat{a}) \sum_{\hat{a}} \nu(\hat{a}|s)[f(s,a,s') + \gamma\hat{V}_f^\nu(s')] \tag{27}$$

$$= \sum_{s'} p(s'|s,a) \sum_{a} \pi(a|\nu(s))[f(s,a,s') + \gamma\hat{V}_f^\nu(s')]. \tag{28}$$

Recall the value function in original safe RL problem,

$$V_f^{\pi\circ\nu}(s) = \sum_{s'} p(s'|s,a) \sum_{a} \pi(a|\nu(s))[f(s,a,s') + \gamma V_f^{\pi\circ\nu}(s')]. \tag{29}$$

Therefore, $V_f^{\pi\circ\nu}(s) = \hat{V}_f^\nu(s), \nu \in \mathcal{N}$. Note that in adversary MDPs $\nu_f^* \in \mathcal{N}$ and

$$\nu_f^* = \arg\max_\nu \mathbb{E}_{a\sim\pi(\cdot|\nu(s)),s'\sim p(\cdot|s,a)}[f(s,a,s') + \gamma\hat{V}_f^\nu(s')]. \tag{30}$$

We also know that ν_f^* is deterministic,

$$\Rightarrow \nu_f^*(s) = \arg\max_\nu \mathbb{E}_{a\sim\pi(\cdot|\tilde{s}),s'\sim p(\cdot|s,a)}[f(s,a,s') + \gamma\hat{V}_f^\nu(s')] \tag{31}$$

$$= \arg\max_\nu \mathbb{E}_{a\sim\pi(\cdot|\tilde{s}),s'\sim p(\cdot|s,a)}[f(s,a,s') + \gamma V_f^{\pi\circ\nu}(s')] \tag{32}$$

$$= \arg\max_\nu V_f^{\pi\circ\nu}(s,a). \tag{33}$$

Therefore, $\nu_r^* = \nu_{\text{MR}}, \nu_c^* = \nu_{\text{MC}}$.

Optimality. We will prove the optimality by contradiction. By definition, $\forall s \in \mathcal{S}$,

$$V_c^{\pi\circ\nu'}(s_0) \leq V_c^{\pi\circ\nu_{\text{MC}}}(s_0). \tag{34}$$

Suppose $\exists\nu'$, $s.t. V_c^{\pi\circ\nu'}(\mu_0) > V_c^{\pi\circ\nu_{\text{MC}}}(\mu_0)$, then there also exists $s_0 \in \mathcal{S}$, $s.t. V_c^{\pi\circ\nu'}(s_0) > V_c^{\pi\circ\nu_{\text{MC}}}(s_0)$, which is contradictory to Eq.(34). Similarly, we can also prove that the property holds for ν_{MR} by replacing $V_c^{\pi\circ\nu}$ with $V_r^{\pi\circ\nu}$. Therefore, there is no other adversary that achieves higher attack effectiveness than ν_{MR} or higher reward stealthiness than ν_{MR}.

A.4. Proof of Theorem 3.10 – one-step attack cost bound

We have

$$V_c^{\pi,\nu}(s) = \mathbb{E}_{a\sim\pi(\cdot|\nu(s)),s'\sim p(\cdot|s,a)}[c(s,a,s') + \gamma V_c^\pi(s')]. \tag{35}$$

By Bellman equation,

$$V_c^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim p(\cdot|s,a)}[c(s,a,s') + \gamma V(s')]. \tag{36}$$

For simplicity, denote $p_{sa}^{s'} = p(s'|s,a)$ and we have

$$\tilde{V}_c^{\pi,\nu}(s) - \tilde{V}_c^\pi(s) = \sum_{a\in\mathcal{A}} \left(\pi(a|\nu(s)) - \pi(a|s) \sum_{s\in\mathcal{S}} p_{sa}^{s'}(c(s,a,s') + \gamma V_c^\pi(s'))\right) \tag{37}$$

$$\leq \left(\sum_{a\in\mathcal{A}} |\pi(a|\nu(s)) - \pi(a|s)|\right) \max_{a\in\mathcal{A}} \sum_{s\in\mathcal{S}} p_{sa}^{s'}(c(s,a,s') + \gamma V_c^\pi(s')). \tag{38}$$

By definition, $D_{\mathrm{TV}}[\pi(\cdot|\nu(s)\|\pi(\cdot|s)] = \frac{1}{2}\sum_{a\in\mathcal{A}}|\pi(a|\nu(s)) - \pi(a|s)|$, and $c(s,a,s') = 0, s' \in \mathcal{S}_c$. Therefore, we have

$$\tilde{V}_c^{\pi,\nu}(s) - V_c^{\pi}(s) \leq 2D_{TV}[\pi(\cdot|\nu(s)\|\pi(\cdot|s)] \max_{a\in\mathcal{A}} \left(\sum_{s\in\mathcal{S}_c} p_{sa}^{s'} c(s,a,s') + \sum_{s\in\mathcal{S}} p_{sa}^{s'} \gamma V_c^{\pi}(s') \right) \tag{39}$$

$$\leq 2L\|\nu(s) - s\|_p \max_{a\in\mathcal{A}} \left(\sum_{s\in\mathcal{S}_c} p_{sa}^{s'} C_m + \sum_{s\in\mathcal{S}} p_{sa}^{s'} \gamma \frac{C_m}{1-\gamma} \right) \tag{40}$$

$$\leq 2L\epsilon \left(p_s C_m + \frac{\gamma C_m}{1-\gamma} \right). \tag{41}$$

A.5. Proof of Theorem 3.11 – episodic attack cost bound

According to the Corollary 2 in CPO (Achiam et al., 2017),

$$V_c^{\pi\circ\nu}(\mu_0) - V_c^{\pi}(\mu_0) \leq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi, a\sim\pi\circ\nu} \left[A_c^{\pi}(s,a) + \frac{2\gamma\delta_c^{\pi\circ\nu}}{1-\gamma} D_{\mathrm{TV}}[\pi'(\cdot|s)\|\pi(\cdot|s)] \right], \tag{42}$$

where $\delta_c^{\pi\circ\nu} = \max_s |\mathbb{E}_{a\sim\pi\circ\nu} A_c^{\pi}(s,a)|$ and $A_c^{\pi}(s,a) = \mathbb{E}_{s'\sim p(\cdot|s,a)}[c(s,a,s') + \gamma V_c^{\pi}(s') - V_c^{\pi}(s)]$ denotes the advantage function. Note that

$$\mathbb{E}_{a\sim\pi\circ\nu} A_c^{\pi}(s,a) = \mathbb{E}_{a\sim\pi\circ\nu}[\mathbb{E}_{s'\sim p(\cdot|s,a)}[c(s,a,s') + \gamma V_c^{\pi}(s') - V_c^{\pi}(s)]] \tag{43}$$

$$= \mathbb{E}_{a\sim\pi\circ\nu, s'\sim p(\cdot|s,a)}[c(s,a,s') + \gamma V_c^{\pi}(s')] - V_c^{\pi}(s) \tag{44}$$

$$= \tilde{V}_c^{\pi,\nu}(s) - V_c^{\pi}(s). \tag{45}$$

By theorem 3.10,

$$\delta_c^{\pi\circ\nu} = \max_s |\mathbb{E}_{a\sim\pi\circ\nu} A_c^{\pi}(s,a)| \tag{46}$$

$$\leq \max_s \left| 2L\epsilon \left(p_s C_m + \frac{\gamma C_m}{1-\gamma} \right) \right| \tag{47}$$

$$= 2L\epsilon C_m \left(\max_s p_s + \frac{\gamma}{1-\gamma} \right). \tag{48}$$

Therefore, we can derive

$$V_c^{\pi\circ\nu}(\mu_0) - V_c^{\pi}(\mu_0) \leq \frac{1}{1-\gamma} \max_s |\mathbb{E}_{a\sim\pi\circ\nu} A_c^{\pi}(s,a)| + \frac{2\gamma\delta_c^{\pi\circ\nu}}{(1-\gamma)^2} D_{\mathrm{TV}}[\pi'(\cdot|s)\|\pi(\cdot|s)] \tag{49}$$

$$= \left(\frac{1}{1-\gamma} + \frac{2\gamma D_{\mathrm{TV}}}{(1-\gamma)^2} \right) \delta_c^{\pi\circ\nu} \tag{50}$$

$$\leq 2L\epsilon C_m \left(\frac{1}{1-\gamma} + \frac{4\gamma L\epsilon}{(1-\gamma)^2} \right) \left(\max_s p_s + \frac{\gamma}{1-\gamma} \right). \tag{51}$$

Note π is a feasible policy, i.e., $V_c^{\pi}(\mu_0) \leq \kappa$. Therefore,

$$V_c^{\pi\circ\nu}(\mu_0) \leq \kappa + 2L\epsilon C_m \left(\frac{1}{1-\gamma} + \frac{4\gamma L\epsilon}{(1-\gamma)^2} \right) \left(\max_s p_s + \frac{\gamma}{1-\gamma} \right). \tag{52}$$

A.6. Proof of Theorem 4.2 – Bellman contraction

Recall Theorem 4.2, the Bellman operators $\mathcal{T}_\pi, \mathcal{T}_c^*, \mathcal{T}_r^*$ are contractions under the sup-norm $\|\cdot\|_\infty$ and will converge to their fixed points, respectively. In addition, the fixed point for $\mathcal{T}_c^*$ is $V_c^{\pi\circ\nu_{\mathrm{MC}}} = \mathcal{T}_c^* V_c^{\pi\circ\nu_{\mathrm{MC}}}$, and the fixed point for $\mathcal{T}_r^*$ is $V_r^{\pi\circ\nu_{\mathrm{MR}}} = \mathcal{T}_r^* V_r^{\pi\circ\nu_{\mathrm{MR}}}$.

Recall the operators definitions, where $B_p^\epsilon(s)$ is the ℓ_p ball constraint with size ϵ:

$$(\mathcal{T}_\pi V_f^{\pi\circ\nu})(s) = \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p(s'|s,a) \left[f(s,a,s') + \gamma V_f^{\pi\circ\nu}(s') \right], \quad f \in \{r, c\}, \tag{53}$$

$$(\mathcal{T}_c^* V_c^{\pi\circ\nu})(s) = \max_{\tilde{s}\in B_p^\epsilon(s)} \sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[c(s,a,s') + \gamma V_c^{\pi\circ\nu}(s')\right], \tag{54}$$

$$(\mathcal{T}_r^* V_r^{\pi\circ\nu})(s) = \max_{\tilde{s}\in B_p^\epsilon(s)} \sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[r(s,a,s') + \gamma V_r^{\pi\circ\nu}(s')\right]. \tag{55}$$

We first prove that the Bellman policy operator $\mathcal{T}_\pi$ is a contraction.

Proof. Denote $f_{sa}^{s'} = f(s,a,s'), f \in \{r,c\}$ and $p_{sa}^{s'} = p(s'|s,a)$ for simplicity, we have:

$$\left|(\mathcal{T}_\pi U_f^{\pi\circ\nu})(s) - (\mathcal{T}_\pi V_f^{\pi\circ\nu})(s)\right| = \Big|\sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[f_{sa}^{s'} + \gamma U_f^{\pi\circ\nu}(s')\right] \tag{56}$$

$$- \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[f_{sa}^{s'} + \gamma V_f^{\pi\circ\nu}(s')\right]\Big| \tag{57}$$

$$= \gamma\Big|\sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[U_f^{\pi\circ\nu}(s') - V_f^{\pi\circ\nu}(s')\right]\Big| \tag{58}$$

$$\leq \gamma \max_{s'\in\mathcal{S}}\left|U_f^{\pi\circ\nu}(s') - V_f^{\pi\circ\nu}(s')\right| \tag{59}$$

$$= \gamma\left\|U_f^{\pi\circ\nu}(s') - V_f^{\pi\circ\nu}(s')\right\|_\infty, \tag{60}$$

Since the above holds for any state s, we have:

$$\max_s \left|(\mathcal{T}_\pi U_f^{\pi\circ\nu})(s) - (\mathcal{T}_\pi V_f^{\pi\circ\nu})(s)\right| \leq \gamma\left\|U_f^{\pi\circ\nu}(s') - V_f^{\pi\circ\nu}(s')\right\|_\infty,$$

which implies that:

$$\left\|(\mathcal{T}_\pi U_f^{\pi\circ\nu})(s) - (\mathcal{T}_\pi V_f^{\pi\circ\nu})(s)\right\|_\infty \leq \gamma\left\|V_f^{\pi\circ\nu_2}(s') - V_f^{\pi\circ\nu_2}(s')\right\|_\infty,$$

Then based on the Contraction Mapping Theorem (Meir & Keeler, 1969), we know that $\mathcal{T}_\pi$ has a unique fixed point $V_f^*(s), f \in \{r,c\}$ such that $V_f^*(s) = (\mathcal{T}_\pi V_f^*)(s)$. □

We then prove that the Bellman adversary effectiveness and stealthiness operators $\mathcal{T}_f^*, f \in \{r,c\}$ is a contraction under the sup-norm $\|\cdot\|_\infty$.

To finish the proof, we first introduce the following lemma:

Lemma A.3. *Suppose* $\max_x h(x) \geq \max_x g(x)$ *and denote* $x^{h*} = \arg\max_x h(x)$*, we have:*

$$\begin{aligned}|\max_x h(x) - \max_x g(x)| &= \max_x h(x) - \max_x g(x) = h(x^{h*}) - \max_x g(x) \\ &\leq h(x^{h*}) - g(x^{h*}) \leq \max_x |h(x) - g(x)|.\end{aligned} \tag{61}$$

Proof.

$$\left|(\mathcal{T}_f^* V_f^{\pi\circ\nu_1})(s) - (\mathcal{T}_f^* V_f^{\pi\circ\nu_2})(s)\right| = \Big|\max_{\tilde{s}\in B_p^\epsilon(s)} \sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[f_{sa}^{s'} + \gamma V_f^{\pi\circ\nu_1}(s')\right] \tag{62}$$

$$- \max_{\tilde{s}\in B_p^\epsilon(s)} \sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[f_{sa}^{s'} + \gamma V_f^{\pi\circ\nu_2}(s')\right]\Big| \tag{63}$$

$$= \Big|\gamma \max_{\tilde{s}\in B_p^\epsilon(s)} \sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[V_f^{\pi\circ\nu_1}(s') - V_f^{\pi\circ\nu_2}(s')\right]\Big| \tag{64}$$

$$\leq \gamma \max_{\tilde{s}\in B_p^\epsilon(s)} \Big|\sum_{a\in\mathcal{A}} \pi(a|\tilde{s}) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[V_f^{\pi\circ\nu_1}(s') - V_f^{\pi\circ\nu_2}(s')\right]\Big| \tag{65}$$

$$\triangleq \gamma\Big|\sum_{a\in\mathcal{A}} \pi(a|\tilde{s}^*) \sum_{s'\in\mathcal{S}} p_{sa}^{s'}\left[V_f^{\pi\circ\nu_1}(s') - V_f^{\pi\circ\nu_2}(s')\right]\Big| \tag{66}$$

$$\leq \gamma \max_{s'\in\mathcal{S}}\left|V_f^{\pi\circ\nu_1}(s') - V_f^{\pi\circ\nu_2}(s')\right| \tag{67}$$

$$= \gamma\left\|V_f^{\pi\circ\nu_1}(s') - V_f^{\pi\circ\nu_2}(s')\right\|_\infty, \tag{68}$$

where inequality (65) comes from Lemma A.3, and $\tilde{s}^*$ in Eq. (66) denote the argmax of the RHS.

Since the above holds for any state s, we can also conclude that:

$$\left\|(\mathcal{T}_f^* V_f^{\pi\circ\nu_1})(s) - (\mathcal{T}_f^* V_f^{\pi\circ\nu_2})(s)\right\|_\infty \leq \gamma \left\|V_f^{\pi\circ\nu_2}(s') - V_f^{\pi\circ\nu_2}(s')\right\|_\infty,$$

□

We then prove that the value function of the MC and MR adversaries $V_c^{\pi\circ\nu_{\text{MC}}}(s), V_r^{\pi\circ\nu_{\text{MR}}}(s)$ are the fixed points for $\mathcal{T}_c^*, \mathcal{T}_r^*$.

Proof. Recall that the MC, MR adversaries are:

$$\nu_{\text{MC}}(s) = \arg\max_{\tilde{s}\in B_p^\epsilon(s)} \mathbb{E}_{\tilde{a}\sim\pi(a|\tilde{s})}\left[Q_c^\pi(s,\tilde{a})\right], \nu_{\text{MR}}(s) = \arg\max_{\tilde{s}\in B_p^\epsilon(s)} \mathbb{E}_{\tilde{a}\sim\pi(a|\tilde{s})}\left[Q_r^\pi(s,\tilde{a})\right]. \tag{69}$$

Based on the value function definition, we have:

$$V_c^{\pi\circ\nu_{\text{MC}}}(s) = \mathbb{E}_{\tau\sim\pi\circ\nu_{\text{MC}}, s_0=s}[\sum_{t=0}^{\infty}\gamma^t c_t] = \mathbb{E}_{\tau\sim\pi\circ\nu_{\text{MC}}, s_0=s}[c_0 + \gamma\sum_{t=1}^{\infty}\gamma^{t-1}c_t] \tag{70}$$

$$= \sum_{a\in\mathcal{A}}\pi(a|\nu_{\text{MC}}(s))\sum_{s'\in\mathcal{S}}p_{sa}^{s'}\left[c(s,a,s') + \gamma\mathbb{E}_{\tau\sim\pi\circ\nu_{\text{MC}}, s_1=s'}[\sum_{t=1}^{\infty}\gamma^{t-1}c_t]\right] \tag{71}$$

$$= \sum_{a\in\mathcal{A}}\pi(a|\nu_{\text{MC}}(s))\sum_{s'\in\mathcal{S}}p_{sa}^{s'}\left[c(s,a,s') + \gamma V_c^{\pi\circ\nu_{\text{MC}}}(s')\right] \tag{72}$$

$$= \max_{\tilde{s}\in B_p^\epsilon(s)}\sum_{a\in\mathcal{A}}\pi(a|\tilde{s})\sum_{s'\in\mathcal{S}}p_{sa}^{s'}\left[c(s,a,s') + \gamma V_c^{\pi\circ\nu_{\text{MC}}}(s')\right] \tag{73}$$

$$= (\mathcal{T}_c^* V_c^{\pi\circ\nu_{\text{MC}}})(s), \tag{74}$$

where Eq. (73) is from the MC attacker definition. Therefore, the cost value function of the MC attacker $V_c^{\pi\circ\nu_{\text{MC}}}$ is the fixed point of the Bellman adversary effectiveness operator $\mathcal{T}_c^*$. With the same procedure (replacing $\nu_{\text{MC}}, \mathcal{T}_c^*$ with $\nu_{\text{MR}}, \mathcal{T}_r^*$), we can prove that the reward value function of the MR attacker $V_r^{\pi\circ\nu_{\text{MR}}}$ is the fixed point of the Bellman adversary stealthiness operator $\mathcal{T}_r^*$.

□

B. Remarks

B.1. Remarks of the safe RL setting and stealthiness

Safe RL setting regarding the reward and the cost. We consider the safe RL problems that have separate task rewards and constraint violation costs, i.e. independent reward and cost functions. Combining the cost with reward to a single scalar metric, which can be viewed as manually selecting Lagrange multipliers, may work in simple problems. However, it lacks interpretability – it is hard to explain what does a single scalar value mean, and requires good domain knowledge of the problem – the weight between costs and rewards should be carefully balanced, which is difficult when the task rewards already contain many objectives/factors. On the other hand, separating the costs from rewards is easy to monitor the safety performance and task performance respectively, which is more interpretable and applicable for different cost constraint thresholds.

(Reward) Stealthy attack for safe RL. As we discussed in Sec. 3.2, the stealthiness concept in supervised learning refers to that the adversarial attack should be covert to prevent from being easily identified. While we use the perturbation set B_p^ϵ to ensure the stealthiness regarding the observation corruption, we notice that another level of stealthiness regarding the task reward performance is interesting and worthy of being discussed. In some real-world applications, the task-related metrics (such as velocity, acceleration, goal distances) are usually easy to be monitored from sensors. However, the safety metrics can be sparse and hard to monitor until breaking the constraints, such as colliding with obstacles and entering hazard states, which are determined by binary indicator signals. Therefore, a dramatic task-related metrics (reward) drop might be easily detected by the agent, while constraint violation signals could be hard to detect until catastrophic failures. An unstealthy attack in this scenario may decrease the reward a lot and prohibit the agent from finishing the task, which can warn the agent that it is attacked and thus lead to a failing attack. On the contrary, a stealthy attack is to maintain the agent's task reward such that the agent is not aware of the existence of the attacks based on "good" task metrics, while performing successful attacks by leading to constraint violations. In other words, a stealthy attack should corrupt the policy to be tempted, since all the tempting policies are high-rewarding while unsafe.

B.2. Remarks of the failure of SA-PPOL(MC/MR) baselines

The detailed algorithm of SA-PPOL (Zhang et al., 2020a) can be found in Appendix C.5. The basic idea can be summarized via the following equation:

$$\ell_\nu(s) = -D_{KL}[\pi(\cdot|s)||\pi_\theta(\cdot|\nu(s))], \tag{75}$$

which aims to minimize the divergence between the corrupted states and the original states. Note that we only optimize (compute gradient) for $\pi_\theta(\cdot|\nu(s))$ rather than $\pi(\cdot|s)$, since we view $\pi(\cdot|s)$ as the "ground-truth" target action distribution. Adding the above KL regularizer to the original PPOL loss yields the SA-PPOL algorithm. We could observe the original SA-PPOL that uses the MAD attacker as the adversary can learn well in most of the tasks, though it is not safe under strong attacks. However, SA-PPOL with MR or MC adversaries often fail to learn a meaningful policy in many tasks, especially for the MR attacker. The reason is that: the MR attacker aims to find the high-rewarding adversarial states, while the KL loss will make the policy distribution of high-rewarding adversarial states to match with the policy distribution of the original relatively lower-rewards states. As a result, the training could fail due to wrong policy optimization direction and prohibited exploration to high-rewarding states. Since the MC attacker can also lead to high-rewarding adversarial states due to the existence of tempting polices, we may also observe failure training with the MC attacker.

C. Implementation Details

C.1. MC and MR attackers implementation

We use the gradient of the state-action value function $Q(s,a)$ to provide the direction to update states adversarially in K steps ($Q = Q_r^\pi$ for MR and $Q = Q_c^\pi$ for MC):

$$s^{k+1} = \text{Proj}[s^k - \eta \nabla_{s^k} Q(s^0, \pi(s^k))], k = 0, \dots, K-1 \tag{76}$$

where Proj$[\cdot]$ is a projection to $B_p^\epsilon(s^0)$, η is the learning rate, and s^0 is the state under attack. Note that we use the gradient of $Q(s^0, \pi(s^k))$ rather than $Q(s^k, \pi(s^k))$ to make the optimization more stable, since the Q function may not generalize well to unseen states in practice. The implementation of MC and MR attacker is shown in algorithm 2.

Algorithm 2 MC and MR attacker

Input: A policy π under attack, corresponding Q networks, initial state s^0, attack steps K, attacker learning rate η, perturbation range ϵ, two thresholds ϵ_Q and ϵ_s for early stopping
Output: An adversarial state $\tilde{s}$

for $k = 1$ to K **do**
  $g^k = \nabla_{s^{k-1}} Q(s_0, \pi(s^{k-1}))$
  $s^k \leftarrow \text{Proj}[s^{k-1} - \eta g^k]$
  Compute $\delta Q = |Q(s_0, \pi(s^k)) - Q(s_0, \pi(s^{k-1}))|$ and $\delta s = |s^k - s^{k-1}|$
  if $\delta Q < \epsilon_Q$ and $\delta s < \epsilon_s$ **then**
    break for early stopping
  end if
end for

C.2. PPO-Lagrangian algorithm

The objective of PPO (clipped) has the form (Schulman et al., 2017):

$$\ell_{ppo} = \min(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \text{clip}(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon) A^{\pi_{\theta_k}}(s,a)) \tag{77}$$

We use PID Lagrangian (Stooke et al., 2020) that addresses the oscillation and overshoot problem in Lagrangian methods. The loss of the PPO-Lagrangian has the form:

$$\ell_{ppol} = \frac{1}{1+\lambda}(\ell_{ppo} + V_r^\pi - \lambda V_c^\pi) \tag{78}$$

The Lagrangian multiplier λ is computed by applying feedback control to V_c^π and is determined by K_P, K_I, and K_D that need to be fine-tuned.

C.3. Adversarial training full algorithm

Due to the page limit, we omit some implementation details in the main content. We will present the full algorithm and some implementation tricks in this section. Without otherwise statement, the critics' and policies' parameterization is assumed to be neural networks (NN), while we believe other parameterization form should also work well.

Critics update. Denote ϕ_r as the parameters for the task reward critic Q_r, and ϕ_c as the parameters for the constraint violation cost critic Q_c. Similar to many other off-policy algorithms (Lillicrap et al., 2015), we use a target network for each critic and the polyak smoothing trick to stabilize the training. Other off-policy critics training methods, such as Re-trace (Munos et al., 2016), could also be easily incorporated with PPO-Lagrangian training framework. Denote ϕ_r' as the parameters for the **target** reward critic Q_r', and ϕ_c' as the parameters for the **target** cost critic Q_c'. Define $\mathcal{D}$ as the replay buffer and (s, a, s', r, c) as the state, action, next state, reward, and cost respectively. The critics are updated by minimizing the following mean-squared Bellman error (MSBE):

$$\ell(\phi_r) = \mathbb{E}_{(s,a,s',r,c)\sim\mathcal{D}}\Big[\left(Q_r(s,a) - (r + \gamma \mathbb{E}_{a'\sim\pi}[Q_r'(s',a')])\right)^2 \Big] \tag{79}$$

$$\ell(\phi_c) = \mathbb{E}_{(s,a,s',r,c)\sim\mathcal{D}}\Big[\left(Q_c(s,a) - (c + \gamma \mathbb{E}_{a'\sim\pi}[Q_c'(s',a')])\right)^2 \Big]. \tag{80}$$

Denote α_c as the critics' learning rate, we have the following updating equations:

$$\phi_r \leftarrow \phi_r - \alpha_c \nabla_{\phi_r} \ell(\phi_r) \tag{81}$$
$$\phi_c \leftarrow \phi_c - \alpha_c \nabla_{\phi_c} \ell(\phi_c) \tag{82}$$

Note that the original PPO-Lagrangian algorithm is an on-policy algorithm, which doesn't require the reward critic and cost critic to train the policy. We learn the critics because the MC and MR attackers require them, which is an essential module for adversarial training.

Polyak averaging for the target networks. The polyak averaging is specified by a weight parameter $\rho \in (0, 1)$ and updates the parameters with:

$$\begin{aligned} \phi'_r &= \rho\phi'_r + (1-\rho)\phi_r \\ \phi'_c &= \rho\phi'_c + (1-\rho)\phi_c \\ \theta' &= \rho\theta' + (1-\rho)\theta. \end{aligned} \tag{83}$$

The critic's training tricks are widely adopted in many off-policy RL algorithms, such as SAC, DDPG and TD3. We observe that the critics trained with those implementation tricks work well in practice. Then we present the full Robust PPO-Lagrangian algorithm:

Algorithm 3 Robust PPO-Lagrangian Algorithm

Input: rollouts T, policy optimization steps M, PPO-Lag loss function $\ell_{ppol}(s, \pi_\theta, r, c)$, adversary function $\nu(s)$, policy parameter θ, critic parameter ϕ_r and ϕ_c, target critic parameter ϕ'_r and ϕ'_c
Output: policy π_θ

Initialize policy parameters and critics parameters
for each training iteration **do**
  Rollout T trajectories by $\pi_\theta \circ \nu$ from the environment $\{(\nu(s), \nu(a), \nu(s'), r, c)\}_N$
  ▷ *Update learner*
  for Optimization steps $m = 1, ..., M$ **do**
    ▷ *No KL regularizer!*
    Compute PPO-Lag loss $\ell_{ppol}(\tilde{s}, \pi_\theta, r, c)$ by Eq. (78)
    Update actor $\theta \leftarrow \theta - \alpha\nabla_\theta \ell_{ppo}$
  end for
  Update value function based on samples $\{(s, a, s', r, c)\}_N$
  ▷ *Update adversary*
  Update critics Q_c and Q_r by Eq. (81) and Eq. (82)
  Polyak averaging target networks by Eq. (83)
  Update adversary based on Q_c and Q_r
end for

C.4. MAD attacker implementation

The full algorithm of MAD attacker is presented in algorithm 4. We use the same SGLD optimizer as in (Zhang et al., 2020a) to maximize the KL-divergence. The objective of the MAD attacker is defined as:

$$\ell_{MAD}(s) = -D_{KL}[\pi(\cdot|s_0)||\pi_\theta(\cdot|s)] \tag{84}$$

Note that we back-propagate the gradient from the corrupted state s instead of the original state s_0 to the policy parameters θ. The full algorithm is shown below:

Algorithm 4 MAD attacker

Input: A policy π under attack, corresponding $Q(s,a)$ network, initial state s^0, attack steps K, attacker learning rate η, the (inverse) temperature parameter for SGLD β, two thresholds ϵ_Q and ϵ_s for early stopping
Output: An adversarial state $\tilde{s}$

for $k = 1$ to K **do**
  Sample $\upsilon \sim \mathcal{N}(0,1)$
  $g^k = \nabla \ell_{MAD}(s_{t-1}) + \sqrt{\frac{2}{\beta\eta}}\upsilon$
  $s^k \leftarrow \text{Proj}[s^{k-1} - \eta g^k]$
  Compute $\delta Q = |Q(s_0, \pi(s^k)) - Q(s_0, \pi(s^{k-1}))|$ and $\delta s = |s^k - s^{k-1}|$
  if $\delta Q < \epsilon_Q$ and $\delta s < \epsilon_s$ **then**
    break for early stopping
  end if
end for

C.5. SA-PPO-Lagrangian baseline

Algorithm 5 SA-PPO-Lagrangian Algorithm

Input: rollouts T, policy optimization steps M, PPO-Lag loss function $\ell_{ppo}(s, \pi_\theta, r, c)$, adversary function $\nu(s)$
Output: policy π_θ

Initialize policy parameters and critics parameters
for each training iteration **do**
  Rollout T trajectories by π_θ from the environment $\{(s, a, s', r, c)\}_N$
  Compute adversary states $\tilde{s} = \nu(s)$ for the sampled trajectories
  ▷ *Update actors*
  for Optimization steps $m = 1, ..., M$ **do**
    Compute KL robustness regularizer $\tilde{L}_{KL} = D_{\text{KL}}(\pi(s) \| \pi_\theta(\tilde{s}))$, no gradient from $\pi(s)$
    Compute PPO-Lag loss $\ell_{ppol}(s, \pi_\theta, r, c)$ by Eq. (78)
    Combine them together with a weight β: $\ell = \ell_{ppol}(s, \pi_\theta, r, c) + \beta\tilde{\ell}_{KL}$
    Update actor $\theta \leftarrow \theta - \alpha\nabla_\theta \ell$
  end for
  ▷ *Update critics*
  Update value function based on samples $\{(s, a, s', r, c)\}_N$
end for

The SA-PPO-Lagrangian algorithm adds an additional KL robustness regularizer to robustify the training policy. Choosing different adversaries ν yields different baseline algorithms. The original SA-PPOL (Zhang et al., 2020a) method adopts the MAD attacker, while we conduct ablation studies by using the MR attacker and the MC attacker, which yields the SA-PPOL(MR) and the SA-PPOL(MC) baselines respectively.

C.6. Improved adaptive MAD (AMAD) attacker baseline

To motivate the design of AMAD baseline, we denote $P^\pi(s'|s) = \int p(s'|s,a)\pi(a|s)da$ as the state transition kernel and $p_t^\pi(s) = p(s_t = s|\pi)$ as the probability of visiting the state s at the time t under the policy π, where $p_t^\pi(s') = \int P^\pi(s'|s)p_{t-1}^\pi(s)ds$. Then the discounted future state distribution $d^\pi(s)$ is defined as (Kakade, 2003):

$$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_t^\pi(s),$$

which allows us to represent the value functions compactly:

$$\begin{aligned} V_f^\pi(\mu_0) &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^\pi, a\sim\pi, s'\sim p}[f(s,a,s')] \\ &= \frac{1}{1-\gamma}\int_{s\in\mathcal{S}} d^\pi(s)\int_{a\in\mathcal{A}}\pi(a|s)\int_{s'\in\mathcal{S}} p(s'|s,a)f(s,a,s')ds'dads, \quad f\in\{r,c\} \end{aligned} \tag{85}$$

Based on Lemma 3.8, the optimal policy π^* in a tempting safe RL setting satisfies:

$$\frac{1}{1-\gamma}\int_{s\in\mathcal{S}} d^{\pi^*}(s)\int_{a\in\mathcal{A}}\pi^*(a|s)\int_{s'\in\mathcal{S}} p(s'|s,a)c(s,a,s')ds'dads = \kappa. \tag{86}$$

We can see that performing MAD attack in low-risk regions that with small $p(s'|s,a)c(s,a,s')$ values may not be effective – the agent may not even be close to the safety boundary. On the other hand, perturbing π when $p(s'|s,a)c(s,a,s')$ is large may have higher chance to result in constraint violations. Therefore, we improve the MAD to the Adaptive MAD attacker, which will only attack the agent in high-risk regions (determined by the cost value function and a threshold ξ).

The implementation of AMAD is shown in algorithm 6. Given a batch of states $\{s\}_N$, we compute the cost values $\{V_c^\pi(s)\}_N$ and sort them in ascending order. Then we select certain percentile of $\{V_c^\pi(s)\}_N$ as the threshold ξ and attack the states that have higher cost value than ξ.

Algorithm 6 AMAD attacker

Input: a batch of states $\{s\}_N$, threshold ξ, a policy π under attack, corresponding $Q(s,a)$ network, initial state s^0, attack steps K, attacker learning rate η, the (inverse) temperature parameter for SGLD β, two thresholds ϵ_Q and ϵ_s for early stopping

Output: batch adversarial state $\tilde{s}$

1: Compute batch cost values $\{V_c^\pi(s)\}_N$
2: $\xi \leftarrow (1-\xi)$ percentile of $V_c^\pi(s)$
3: **for** the state s that $V_c^\pi(s) > \xi$ **do**
4: compute adversarial state $\tilde{s}$ by algorithm 4
5: **end for**

C.7. Environment description

We use the Bullet safety gym (Gronauer, 2022) environments for this set of experiments. In the Circle tasks, the goal is for an agent to move along the circumference of a circle while remaining within a safety region smaller than the radius of the circle. The reward and cost functions are defined as:

$$\begin{aligned} r(s) &= \frac{-yv_x + xv_y}{1+|\sqrt{x^2+y^2}-r|} + r_{robot}(s) \\ c(s) &= \mathbf{1}(|x| > x_{lim}) \end{aligned}$$

where x, y are the position of the agent on the plane, v_x, v_y are the velocities of the agent along the x and y directions, r is the radius of the circle, and x_{lim} specified the range of the safety region, $r_{robot}(s)$ is the specific reward for different robot. For example, an ant robot will gain reward if its feet do not collide with each other. In the Run tasks, the goal for an agent is to move as far as possible within the safety region and the speed limit. The reward and cost functions are defined as:

$$\begin{aligned} r(s) &= \sqrt{(x_{t-1}-g_x)^2-(y_{t-1}-g_y)^2} - \sqrt{(x_t-g_x)^2-(y_t-g_y)^2} + r_{robot}(s) \\ c(s) &= \mathbf{1}(|y| > y_{lim}) + \mathbf{1}(\sqrt{v_x^2+v_y^2} > v_{lim}) \end{aligned}$$

where v_{lim} is the speed limit and g_x and g_y is the position of a fictitious target. The reward is the difference between current distance to the target and the distance in the last timestamp.

C.8. Hyper-parameters

In all experiments, we use Gaussian policies with mean vectors given as the outputs of neural networks, and with variances that are separate learnable parameters. The policy networks and Q networks for all experiments have two hidden layers of sizes (256, 256) with ReLU activation functions. We use a discount factor of $\gamma = 0.995$, a GAE-λ for estimating the regular advantages of $\lambda^{GAE} = 0.97$, a KL-divergence step size of $\delta_{KL} = 0.01$, a clipping coefficient of 0.02. The PID parameters for the Lagrange multiplier are: $K_p = 0.1$, $K_I = 0.003$, and $K_D = 0.001$. The learning rate of the adversarial attackers: MAD, AMAD, MC, and MR is 0.05. The optimization steps of MAD and AMAD is 60 and 200 for MC and MR attacker. The threshold ξ for AMAD is 0.1. The complete hyperparameters used in the experiments are shown in Table 2. We choose larger perturbation range for the Car robot-related tasks because they are simpler and easier to train.

Table 2: Hyperparameters for all the environments

Parameter	Car-Run	Dron-Run	Ant-Run	Car-Circle	Dron-Circle	Ant-Circle
training epoch	100	250	250	500	500	1000
batch size	10000	20000	20000	15000	15000	30000
minibatch size	300	300	300	300	300	300
rollout length	200	100	200	300	300	300
cost limit	5	5	5	5	5	5
perturbation ϵ	0.05	0.025	0.025	0.05	0.025	0.025
actor optimization step M	80	80	80	80	80	160
actor learning rate	0.0003	0.0002	0.0005	0.0003	0.0003	0.0005
critic learning rate	0.001	0.001	0.001	0.001	0.001	0.001

C.9. More experiment results

All the experiments are performed on a server with AMD EPYC 7713 64-Core Processor CPU. For each experiment, we use 4 CPUs to train each agent that is implemented by PyTorch, and the training time varies from 4 hours (Car-Run) to 3 days (Ant-Circle). Video demos are available at: https://sites.google.com/view/robustsaferl/home

Here we evaluate the performance of MAD and AMAD adversaries by attacking well-trained PPO-Lagrangian policies in Car-Run and Ant-Run task. We keep the policies' model weights fixed for all the attackers for fair comparison. The comparison is shown in Fig. 3. We vary the attacking fraction (determined by ξ) to thoroughly study the effectiveness of the AMAD attacker. We can see that AMAD attacker is more effective because the cost increases significantly with the increase in perturbation, while the reward is maintained well. This validates our hypothesis that attacking the agent in high-risk regions is more effective and stealthy.

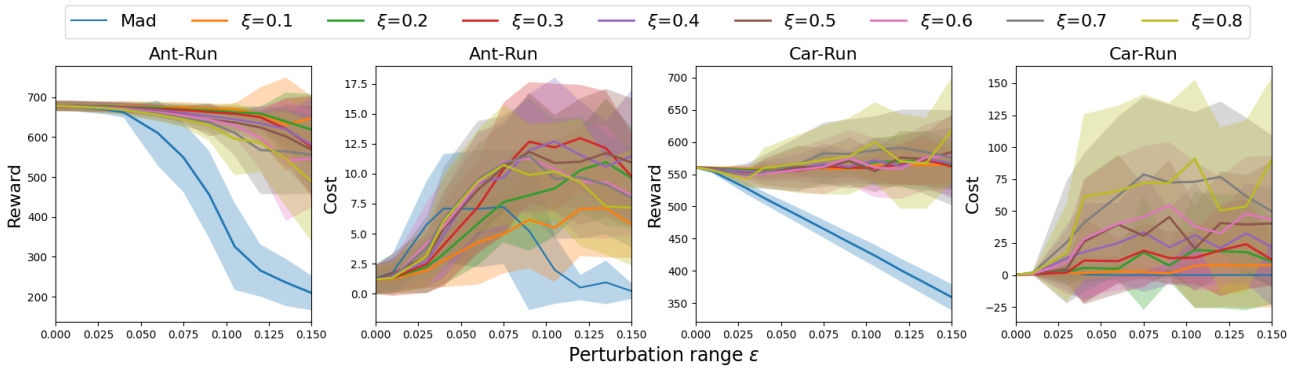


Figure 3: Reward and cost of AMAD and MAD attacker

The experiment results of trained safe RL policies under the Random and MAD attackers are shown in Table 3. The last column shows the average rewards and costs over all the 5 attackers (Random, MAD, AMAD, MC, MR). Our agent (ADV-PPOL) with adversarial training is robust against all the 5 attackers and achieves the lowest cost. We can also see that AMAD attacker is more effective than MAD since the cost under the AMAD attacker is higher than the cost under the MAD attacker.

Table 3: Evaluation results of natural performance (no attack) and under Random and MAD attackers. The average column shows the average rewards and costs over all 5 attackers (Random, MAD, AMAD, MC, and MR). Our methods are ADV-PPOL(MC/MR). Each value is reported as: mean ± standard deviation for 50 episodes and 5 seeds. We shadow two lowest-costs agents under each attacker column and break ties based on rewards, excluding the failing agents (whose natural rewards are less than 50% of PPOL-vanilla's. We mark the failing agents with ⋆.

Env	Method	Random		MAD		Average	
		Reward	Cost	Reward	Cost	Reward	Cost
Car-Circle $\epsilon = 0.05$	PPOL-vanilla	328.34±118.08	20.67±18.65	178.21±81.31	28.27±49.12	280.42±87.0	42.41±13.74
	PPOL-random	393.49±43.87	2.17±3.44	272.0±75.21	81.63±43.47	337.16±51.76	37.65±12.91
	SA-PPOL(MAD)	397.71±20.87	0.13±0.5	366.64±25.82	0.93±2.73	394.92±15.33	28.57±1.75
	SA-PPOL(MC)	383.92±19.5	0.0±0.0	288.28±25.2	0.0±0.0	376.5±7.09	26.61±2.39
	SA-PPOL(MR)	370.71±47.18	0.54±1.94	308.8±22.49	0.1±0.7	366.16±37.08	22.08±4.6
	ADV-PPOL(MC)	302.61±11.81	0.0±0.0	292.83±23.04	2.22±4.75	295.46±8.04	0.83±1.35
	ADV-PPOL(MR)	309.81±34.96	0.0±0.0	312.18±15.81	8.76±11.52	304.34±10.29	2.24±3.1
Drone-Circle $\epsilon = 0.025$	PPOL-vanilla	603.53±85.34	6.5±7.03	469.47±186.11	69.17±38.21	452.41±51.12	54.57±19.34
	PPOL-random	585.71±108.76	6.87±33.14	456.66±155.61	58.6±42.59	442.21±44.87	41.21±24.18
	SA-PPOL(MAD)	500.49±18.23	0.0±0.0	491.23±25.15	0.23±0.96	467.1±54.85	29.61±12.62
	SA-PPOL(MC)	357.65±49.52	0.0±0.0	343.52±50.41	0.47±1.77	352.13±49.02	20.83±10.19
	⋆ SA-PPOL(MR)	187.81±129.74	19.18±53.92	180.62±122.42	15.06±41.66	191.66±123.83	23.4±21.63
	ADV-PPOL(MC)	359.45±26.63	0.0±0.0	325.92±46.12	4.22±13.82	358.74±35.95	2.79±4.97
	ADV-PPOL(MR)	352.77±51.5	0.0±0.0	331.06±63.45	4.4±14.96	341.8±37.13	2.37±6.06
Ant-Circle $\epsilon = 0.025$	PPOL-vanilla	152.98±21.02	0.9±3.59	157.36±22.76	5.27±10.27	160.93±17.15	17.69±7.0
	PPOL-random	159.02±23.93	3.13±8.15	155.34±27.44	2.8±5.47	153.63±13.59	10.56±4.41
	SA-PPOL(MAD)	140.21±39.95	4.6±21.18	146.38±34.43	1.47±5.19	152.47±22.08	14.4±8.09
	⋆ SA-PPOL(MC)	-0.38±1.57	0.0±0.0	-0.73±1.88	0.0±0.0	-0.3±0.8	0.0±0.0
	⋆ SA-PPOL(MR)	-0.53±2.07	0.0±0.0	-0.89±2.25	0.0±0.0	-0.66±1.09	0.0±0.0
	ADV-PPOL(MC)	131.22±18.72	0.3±1.29	132.95±18.85	0.03±0.18	132.55±14.1	1.86±2.51
	ADV-PPOL(MR)	126.91±23.59	0.73±2.93	134.82±19.38	1.63±4.37	131.35±13.22	1.02±1.56
Car-Run $\epsilon = 0.05$	PPOL-vanilla	553.61±2.81	19.47±6.19	504.29±9.71	0.49±5.94	567.75±3.38	58.84±4.68
	PPOL-random	555.24±1.89	0.92±1.17	542.84±2.2	2.61±2.44	561.68±1.52	53.28±0.49
	SA-PPOL(MAD)	545.86±2.11	0.0±0.0	548.11±2.2	0.0±0.0	553.52±1.62	1.17±1.31
	SA-PPOL(MC)	540.36±2.83	0.0±0.0	522.8±3.1	0.0±0.0	549.06±2.55	0.26±0.44
	SA-PPOL(MR)	539.04±1.31	0.0±0.0	529.38±1.91	0.0±0.0	546.74±1.12	4.07±5.73
	ADV-PPOL(MC)	521.85±3.2	0.0±0.0	504.25±4.23	0.0±0.0	528.93±2.74	0.01±0.04
	ADV-PPOL(MR)	522.15±2.31	0.0±0.0	504.16±3.12	0.0±0.0	529.41±2.36	0.02±0.05
Drone-Run $\epsilon = 0.025$	PPOL-vanilla	346.59±2.93	17.33±12.63	348.19±33.21	41.96±28.11	347.52±5.44	37.31±5.46
	PPOL-random	342.68±3.16	3.72±5.6	269.88±14.33	1.66±8.4	321.03±8.03	15.11±6.72
	SA-PPOL(MAD)	306.73±20.71	1.9±4.8	323.19±24.66	29.19±23.81	296.86±53.06	25.79±5.5
	⋆ SA-PPOL(MC)	151.69±19.97	0.01±0.16	77.66±49.01	0.0±0.0	155.15±11.8	2.67±2.11
	⋆ SA-PPOL(MR)	0.09±0.29	0.0±0.0	0.05±0.28	0.0±0.0	0.15±0.29	0.0±0.0
	ADV-PPOL(MC)	277.23±6.89	0.0±0.0	264.26±12.98	0.12±0.69	275.86±9.33	2.77±3.78
	ADV-PPOL(MR)	235.17±20.24	0.0±0.0	230.04±24.4	0.0±0.0	233.9±25.83	1.11±1.29
Ant-Run $\epsilon = 0.025$	PPOL-vanilla	676.14±12.12	1.89±1.77	672.38±12.71	3.59±2.66	678.37±13.99	27.09±5.09
	PPOL-random	671.8±14.45	1.52±1.2	667.73±13.6	2.09±1.43	669.79±8.59	14.37±2.12
	SA-PPOL(MAD)	659.01±13.66	0.55±0.8	658.28±13.9	0.75±0.98	665.21±9.41	22.52±4.88
	SA-PPOL(MC)	575.82±27.89	3.44±3.51	572.12±27.95	3.17±3.47	584.98±25.62	10.06±4.23
	⋆ SA-PPOL(MR)	68.46±93.11	5.27±4.46	77.65±79.75	5.17±4.5	77.83±67.63	5.58±4.34
	ADV-PPOL(MC)	599.93±18.22	0.0±0.0	597.65±18.61	0.0±0.0	618.73±17.7	0.41±0.24
	ADV-PPOL(MR)	618.62±25.38	0.31±0.6	615.31±23.5	0.41±0.68	625.14±21.95	1.46±0.74